

Kurosh Madani
António Dourado Correia
Agostinho Rosa
Joaquim Filipe (Eds.)

Computational Intelligence

Revised and Selected Papers of the
International Joint Conference, IJCCI 2010
Valencia, Spain, October 2010

Kurosh Madani, António Dourado Correia, Agostinho Rosa, and Joaquim Filipe (Eds.)

Computational Intelligence

Studies in Computational Intelligence, Volume 399

Editor-in-Chief

Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series can be found on our homepage:
springer.com

Vol. 379. Ferrante Neri, Carlos Cotta, and
Pablo Moscato (Eds.)
Handbook of Memetic Algorithms, 2011
ISBN 978-3-642-23246-6

Vol. 380. Anthony Brabazon, Michael O'Neill, and
Dietmar Maringer (Eds.)
Natural Computing in Computational Finance, 2011
ISBN 978-3-642-23335-7

Vol. 381. Radosław Katarzyniak, Tzu-Fu Chiu,
Chao-Fu Hong, and Ngoc Thanh Nguyen (Eds.)
*Semantic Methods for Knowledge Management and
Communication*, 2011
ISBN 978-3-642-23417-0

Vol. 382. F.M.T. Brazier, Kees Nieuwenhuis, Gregor Pavlin,
Martijn Warmier, and Costin Badica (Eds.)
Intelligent Distributed Computing V, 2011
ISBN 978-3-642-24012-6

Vol. 383. Takayuki Ito, Minjie Zhang, Valentin Robu,
Shaheen Fatima, and Tokuro Matsuo (Eds.)
New Trends in Agent-Based Complex Automated Negotiations,
2012
ISBN 978-3-642-24695-1

Vol. 384. Daphna Weinshall, Jörn Anemüller,
and Luc van Gool (Eds.)
Detection and Identification of Rare Audiovisual Cues, 2012
ISBN 978-3-642-24033-1

Vol. 385. Alex Graves
Supervised Sequence Labelling with Recurrent Neural Networks,
2012
ISBN 978-3-642-24796-5

Vol. 386. Marek R. Ogiela and Lakhmi C. Jain (Eds.)
*Computational Intelligence Paradigms in Advanced Pattern
Classification*, 2012
ISBN 978-3-642-24048-5

Vol. 387. David Alejandro Pelta, Natalio Krasnogor,
Dan Dumitrescu, Camelia Chira, and Rodica Lung (Eds.)
*Nature Inspired Cooperative Strategies for Optimization (NICSO
2011)*, 2011
ISBN 978-3-642-24093-5

Vol. 388. Tiansi Dong
Recognizing Variable Environments, 2012
ISBN 978-3-642-24057-7

Vol. 389. Patricia Melin
*Modular Neural Networks and Type-2 Fuzzy Systems for Pattern
Recognition*, 2012
ISBN 978-3-642-24138-3

Vol. 390. Robert Bembienik, Lukasz Skonieczny,
Henryk Rybiński, and Marek Niezgodka (Eds.)
Intelligent Tools for Building a Scientific Information Platform,
2012
ISBN 978-3-642-24808-5

Vol. 391. Herwig Unger, Kyandoghene Kyamaky,
and Janusz Kacprzyk (Eds.)
Autonomous Systems: Developments and Trends, 2012
ISBN 978-3-642-24805-4

Vol. 392. Narendra Chauhan, Machavaram Kartikeyan,
and Ankush Mittal
*Soft Computing Methods for Microwave and Millimeter-Wave
Design Problems*, 2012
ISBN 978-3-642-25562-5

Vol. 393. Hung T. Nguyen, Vladik Kreinovich, Berlin Wu,
and Gang Xiang
*Computing Statistics under Interval and Fuzzy
Uncertainty*, 2012
ISBN 978-3-642-24904-4

Vol. 394. David A. Elizondo, Agustí Solanas,
and Antoni Martínez-Ballesté (Eds.)
*Computational Intelligence for Privacy
and Security*, 2012
ISBN 978-3-642-25236-5

Vol. 395. Srikanta Patnaik and Yeon-Mo Yang (Eds.)
Soft Computing Techniques in Vision Science, 2012
ISBN 978-3-642-25506-9

Vol. 396. Marielba Zacarias and
José Valente de Oliveira (Eds.)
Human-Computer Interaction: The Agency Perspective, 2012
ISBN 978-3-642-25690-5

Vol. 397. Elena Nikolaevskaya, Alexandr Khimich,
and Tamara Chistyakova
Programming with Multiple Precision, 2012
ISBN 978-3-642-25672-1

Vol. 398. Fabrice Guillet, Gilbert Ritschard,
and Djamel Abdelkader Zighed (Eds.)
Advances in Knowledge Discovery and Management, 2012
ISBN 978-3-642-25837-4

Vol. 399. Kurosh Madani, António Dourado Correia,
Agostinho Rosa, and Joaquim Filipe (Eds.)
Computational Intelligence, 2012
ISBN 978-3-642-27533-3

Kurosh Madani, António Dourado Correia,
Agostinho Rosa, and Joaquim Filipe (Eds.)

Computational Intelligence

Revised and Selected Papers of the International Joint
Conference, IJCCI 2010, Valencia, Spain, October 2010

Editors

Prof. Kurosh Madani
University PARIS-EST Creteil (UPEC)
Images, Signals and Intelligence
Systems Laboratory
Paris
France

Prof. Agostinho Rosa
Instituto Superior Tecnico IST
Systems and Robotics Institute
Evolutionary Systems and
Biomedical Engineering Lab
Lisboa
Portugal

Prof. António Dourado Correia
University of Coimbra
Departamento de Engenharia
Informatica
Coimbra
Portugal

Prof. Joaquim Filipe
Polytechnic Institute of
Setúbal / INSTICC
Setubal
Portugal

ISSN 1860-949X

e-ISSN 1860-9503

ISBN 978-3-642-27533-3

e-ISBN 978-3-642-27534-0

DOI 10.1007/978-3-642-27534-0

Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012930483

© Springer-Verlag Berlin Heidelberg 2012

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The present book includes extended and revised versions of a set of selected papers from the Second International Joint Conference on Computational Intelligence (IJCCI 2010), held in Valencia, Spain, from 24 to 26 October, 2010.

The purpose of IJCCI is to bring together researchers, engineers and practitioners in computational technologies, especially those related to the areas of fuzzy computation, evolutionary computation and neural computation. IJCCI is composed of three co-located conferences, each one specialized in one of the aforementioned - knowledge areas. Namely:

- International Conference on Fuzzy Computation
- International Conference on Evolutionary Computation
- International Conference on Neural Computation

Their aim is to provide major forums for scientists, engineers and practitioners interested in the study, analysis, design and application of systems.

In the International Conference on Fuzzy Computation (ICFC), modelling and implementation of fuzzy systems, both theoretically and in a broad range of application fields is the main concern. Fuzzy computation is a field that encompasses the theory and application of fuzzy sets and fuzzy logic to the solution of information processing and system analysis problems. Bolstered by information technology developments, the extraordinary growth of fuzzy computation in recent years has led to major applications in fields ranging from medical diagnosis and automated learning to image understanding and systems control.

In the International Conference on Evolutionary Computation (ICEC) modelling and implementation of bioinspired systems namely on the evolutionary premises, both theoretically and in a broad range of application fields, is the central scope. Considered a subfield of computational intelligence focused on combinatorial optimization problems, evolutionary computation is associated with systems that use computational models of evolutionary processes as the key elements in design and implementation, i.e. computational techniques which are inspired by the evolution of biological life in the natural world. A number of evolutionary computational models have been proposed, including evolutionary algorithms, genetic algorithms, the evolution strategy, evolutionary programming, swarm optimization and artificial life.

The International Conference on Neural Computation (ICNC) is focused on modelling and implementation of neural computing systems, both theoretically and also in a broad range of application fields. Neural computation and artificial neural networks have seen an explosion of interest over the last few years, and are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics, in problems of prediction, classification or control. Several architectures, learning strategies and algorithms have been introduced in this highly dynamic field in the last couple of decades.

IJCCI has received 236 paper submissions from 49 countries in all continents. 30 papers were published and presented as full papers, i.e. completed work, 53 papers reflecting work-in-progress or position papers were accepted for short presentation, and another 30 contributions were accepted for poster presentation. These numbers, leading to a “full-paper” acceptance close to 13% and a total oral paper presentations acceptance ratio of about 35%, show the high quality of this forum, to be preserved in the next editions of this conference. This book includes revised and extended versions of a strict selection of the best papers presented at the conference.

Furthermore, IJCCI 2010 included 4 plenary keynote lectures given by James Bezdek, Antonio Sala, Simon M. Lucas and Panos Pardalos. We would like to express our appreciation to all of them and in particular to those who took the time to contribute with a paper to this book.

On behalf of the Conference Organizing Committee, we would like to thank all participants. First of all to the authors, whose quality work is the essence of the conference and to the members of the Program Committee, who helped us with their expertise and diligence in reviewing the papers. As we all know, producing a post-conference book, within the high technical level exigency, requires the effort of many individuals. We wish to thank also all the members of our Organizing Committee, whose work and commitment were invaluable.

September 2011

Kurosh Madani
António Dourado Correia
Agostinho Rosa
Joaquim Filipe

Conference Committee

Conference Co-chairs

Joaquim Filipe Polytechnic Institute of Setúbal / INSTICC,
Portugal
Janusz Kacprzyk Polish Academy of Sciences, Poland

Program Co-chairs

ICEC

Agostinho Rosa IST, Portugal

ICFC

António Dourado Correia University of Coimbra, Portugal

ICNC

Kurosh Madani University PARIS-EST Creteil (UPEC), France

Organizing Committee

Sérgio Brissos INSTICC, Portugal
Helder Coelhas INSTICC, Portugal
Vera Coelho INSTICC, Portugal
Andreia Costa INSTICC, Portugal
Patrícia Duarte INSTICC, Portugal
Bruno Encarnação INSTICC, Portugal
Liliana Medina INSTICC, Portugal
Elton Mendes INSTICC, Portugal
Carla Mota INSTICC, Portugal
Raquel Pedrosa INSTICC, Portugal
Vitor Pedrosa INSTICC, Portugal
Daniel Pereira INSTICC, Portugal
Filipa Rosa INSTICC, Portugal
José Varela INSTICC, Portugal
Pedro Varela INSTICC, Portugal

ICEC Program Committee

Christos Ampatzis, Belgium
Mikhail Prokopenko, Australia
Alice Smith, USA
Michal Bidlo, Czech Republic
Maria J. Blesa, Spain
Christian Blum, Spain
Indranil Bose, China
David Cairns, UK
Rachel Cavill, UK
Ying-ping Chen, Taiwan
Hui Cheng, UK
Chi Kin Chow, China
Leandro dos Santos Coelho, Brazil
Fernando Almeida e Costa, UK
Bernabé Dorronsoro Díaz, Luxembourg
Liliana Dobrica, Romania
Benjamin Doerr, Germany
Jan Drugowitsch, USA
Peter Duerr, Switzerland
Marc Ebner, Germany
Bruce Edmonds, UK
Fabio Fassetti, Italy
Stefka Fidanova, Bulgaria
Dalila Fontes, Portugal
Fabricio Olivetti de França, Brazil
Ozlem Garibay, USA
Carlos Gershenson, Mexico
Narzisi Giuseppe, USA
Daniel Große, Germany
Jörg Hähner, Germany
Christian Haubelt, Germany
Andreas Herkersdorf, Germany
J. Ignacio Hidalgo, Spain
Jeffrey Horn, USA
Enda Howley, Ireland
Jinglu Hu, Japan
Colin Johnson, UK
Mark Johnston, USA
Tatiana Kalganova, UK
Marta Kasprzak, Poland
Ed Keedwell, UK
Mario Köppen, Japan
Karl-Heinz Krempels, Germany
Antonio J. Fernández Leiva, Spain
Piotr Lipinski, Poland
Francisco Luna, Spain
Evelyne Lutton, France
Barry McMullin, Ireland
Jörn Mehnen, UK
Zbigniew Michalewicz, Australia
Luiza de Macedo Mourelle, Brazil
Giuseppe Nicosia, Italy
Schütze Oliver, Mexico
Pietro S. Oliveto, UK
Beatrice Ombuki-Berman, Canada
Ender Özcan, UK
Gary Parker, USA
Petrica Pop, Romania
Aurora Pozo, Brazil
Joaquim Reis, Portugal
Andri Riid, Estonia
Mateen Rizki, USA
Emmanuel Sapin, France
Lukáš Sekanina, Czech Republic
Adam Slowik, Poland
Giandomenico Spezzano, Italy
Sergiu Stan, Romania
Giovanni Straquadanio, Italy
Emilia Tantar, France
Jonathan Thompson, UK
Vito Trianni, Italy
Krzysztof Trojanowski, Poland
Yuan-Jye Tseng, Taiwan
Elio Tuci, Italy
Neal Wagner, Australia
Peter Whigham, New Zealand
Bart Wyns, Belgium
Shiu Yin Yuen, China

ICEC Auxiliary Reviewers

Arjun Chandra, UK
 Matthias Hoffacker, Germany
 Jiri Jaros, Czech Republic
 Pedro Faria Lopes, Portugal

Monica Lora, Germany
 Eddy Parkinson, Australia
 Alexandru-Adrian Tantar, Luxembourg
 Walter Unger, Germany

ICFC Program Committee

Valentina E. Balas, Romania
 Sansanee Auephanwiriyaikul, Thailand
 Ulrich Bodenhofer, Austria
 Jinhai Cai, Australia
 Heloisa Camargo, Brazil
 Martina Dankova, Czech Republic
 Bijan Davvaz, Iran, Islamic Republic of
 Kudret Demirli, Canada
 Ioan Despi, Australia
 Nauck Detlef, UK
 Girolamo Fornarelli, Italy
 Jonathan Garibaldi, UK
 Alexander Gegov, UK
 Susana Muñoz Hernández, Spain
 Zeng-Guang Hou, China
 Angel A. Juan, Spain
 Donald H. Kraft, USA
 Piotr Kulczycki, Poland
 Anne Laurent, France
 Chin-Teng Lin, Taiwan
 Tsung-Chih Lin, Taiwan
 Feilong Liu, USA
 Francesco Marcelloni, Italy
 Francesco Masulli, Italy
 Radko Mesiar, Slovak Republic

Javier Montero, Spain
 Hiroshi Nakajima, Japan
 Mirko Navara, Czech Republic
 Yusuke Nojima, Japan
 Sanja Petrovic, UK
 David Picado, Austria
 Valentina Plekhanova, UK
 Antonello Rizzi, Italy
 Julio Rojas-Mora, Spain
 Mehdi Roopaei, Iran,
 Islamic Republic of
 Alessandra Russo, UK
 Steven Schockaert, Belgium
 Woei Wan Tan, Singapore
 Dat Tran, Australia
 Eiji Uchino, Japan
 José Luis Verdegay, Spain
 Christian Wagner, UK
 Thomas Whalen, USA
 Dongrui Wu, USA
 Chung-Hsing Yeh, Australia
 Jianqiang Yi, China
 Tina Yu, Canada
 Xiao-Jun Zeng, UK
 Hans-Jürgen Zimmermann, Germany

ICFC Auxiliary Reviewer

Luke Dickens, UK

ICNC Program Committee

Veronique Amarger, France
 Ammar Belatreche, UK
 Daniel Berrar, Japan
 Samia Bouchafa, France

Ivo Bukovsky, Czech Republic
 María José Castro-Bleda, Spain
 João Catalão, Portugal
 Ning Chen, Portugal

Amine Chohra, France
Catalina Cocianu, Romania
José Alfredo Ferreira Costa, Brazil
Khalifa Djemal, France
Péter Érdi, USA
Jose M. Ferrandez, Spain
Marcos Gestal, Spain
Vladimir Golovko, Russian Federation
Maria Del Carmen Hernandez Gomez,
Spain
Manuel Grana, Spain
Randa Herzallah, Jordan
Tom Heskens, The Netherlands
Chris Hinde, UK
Robert Hiromoto, USA
Magnus Johnsson, Sweden
Juha Karhunen, Finland
Christel Kemke, Canada
DaeEun Kim, Korea, Republic of
Ekaterina Komendantskaya, UK
Dalia Kriksciuniene, Lithuania
Adam Krzyzak, Canada
Noel Lopes, Portugal
Jinhu Lu, China
Hichem Maaref, France
Kurosh Madani, France
Mitsuharu Matsumoto, Japan
Ali Minai, USA

Adnan Abou Nabout, Germany
João Neto, Portugal
Seiichi Ozawa, Japan
Eliano Pessa, Italy
Manuel Roveri, Italy
Christophe Sabourin, France
Abdel-badeeh Salem, Egypt
Gerald Schaefer, UK
Alon Schclar, Israel
Christoph Schommer, Luxembourg
María Teresa García Sebastián, Spain
Moustapha Séne, Senegal
Shiliang Sun, China
Norikazu Takahashi, Japan
Yi Tang, China
Jim Torresen, Norway
Carlos M. Travieso, Spain
Andrei Utkin, Portugal
Brijesh Verma, Australia
Ricardo Vigario, Finland
Eva Volna, Czech Republic
Fei Wang, USA
Shandong Wu, USA
Pingkun Yan, USA
Cleber Zanchettin, Brazil
Huiyu Zhou, UK

ICNC Auxiliary Reviewers

Moustapha Séne, Senegal
Shiliang Sun, China
Norikazu Takahashi, Japan
Yi Tang, China
Jim Torresen, Norway
Carlos M. Travieso, Spain
Andrei Utkin, Portugal
Brijesh Verma, Australia

Ricardo Vigario, Finland
Eva Volna, Czech Republic
Fei Wang, USA
Shandong Wu, USA
Pingkun Yan, USA
Cleber Zanchettin, Brazil
Huiyu Zhou, UK

Invited Speakers

James Bezdek
Antonio Sala
Simon M. Lucas
Panos Pardalos

University of Melbourne, Australia
Technical University Valencia, Spain
University of Essex, UK
University of Florida, USA

Contents

Invited Paper

- Incremental Kernel Fuzzy c -Means** 3
Timothy C. Havens, James C. Bezdek, Marimuthu Palaniswami

Part I: Evolutionary Computation

- Ant Algorithm for Optimal Sensor Deployment** 21
Stefka Fidanova, Pencho Marinov, Enrique Alba
- Countering Evolutionary Forgetting in No-Limit Texas Hold'em Poker Agents** 31
Garrett Nicolai, Robert Hilderman
- Model Regularization in Coevolutionary Architectures Evolving Straight Line Code** 49
César L. Alonso, José Luis Montaña, Cruz Enrique Borges, Marina de la Cruz Echeandía, Alfonso Ortega de la Puente
- Evolution of Collective Perception in a Group of Autonomous Robots** 67
Giuseppe Morlino, Vito Trianni, Elio Tuci
- Solving SONET Problems Using a Hybrid Scatter Search Algorithm** 81
Anabela Moreira Bernardino, Eugénia Moreira Bernardino, Juan Manuel Sánchez-Pérez, Juan Antonio Gómez-Pulido, Miguel Angel Vega-Rodríguez
- Investigating a Measure of the Recombinational Distance Traversed by the Genetic Algorithm** 99
Robert Collier, Mark Wineberg

Enhancing the Adaptive Dissortative Mating Genetic Algorithm in Fast Non-stationary Fitness Functions 115
Carlos M. Fernandes, Juan Julián Merelo, Agostinho C. Rosa

A Receding Horizon Genetic Algorithm for Dynamic Resource Allocation: A Case Study on Optimal Positioning of Tugs 131
Robin T. Bye

Part II: Fuzzy Computation

Generating Optimized Fuzzy Partitions to Classification and Considerations to Management Imprecise Data 151
J.M. Cadenas, M.C. Garrido, R. Martínez

A Fuzzy Logic Based Approach to Expressing and Reasoning with Uncertain Knowledge on the Semantic Web 167
Jidi Zhao, Harold Boley, Weichang Du

Portfolio Investment Decision Support System Based on a Fuzzy Inference System 183
Isidoro J. Casanova

Fuzzy Analytical Network Models for Metasearch 197
Arijit De, Elizaebeth Diaz

On the Satisfiability and Validity Problems in the Propositional Gödel Logic 211
Dušan Guller

A Fuzzy Approach to Resource Aware Automatic Parallelization 229
T. Trigo de la Vega, P. Lopez-Garcia, S. Muñoz-Hernández

Fuzzy and Fractal Technology in Market Analysis 247
Petr Kroha, Marcus Lauschke

The Banach Contraction Principle in Fuzzy Quasi-metric Spaces and in Product Complexity Spaces: Two Approaches to Study the Cost of Algorithms with a Finite System of Recurrence Equations 261
Francisco Castro-Company, Salvador Romaguera, Pedro Tirado

Part III: Neural Computation

SVM-Based Object Detection Using Self-quotient ε -Filter and Histograms of Oriented Gradients 277
Mitsuharu Matsumoto

Adaptive Control of Robot Systems with Simple Rules Using Chaotic Dynamics in Quasi-layered Recurrent Neural Networks	287
<i>Ryosuke Yoshinaka, Masato Kawashima, Yuta Takamura, Hitoshi Yamaguchi, Naoya Miyahara, Kei-ichiro Nabeta, Yongtao Li, Shigetoshi Nara</i>	
Mathematical Modeling of Human Thermoregulation: A Neurophysiological Approach to Vasoconstriction	307
<i>Boris R.M. Kingma, Arjan J.H. Frijns, Wim H. Saris, Anton A. van Steenhoven, Wouter D. van Marken Lichtenbelt</i>	
Visual Target Selection Emerges from a Bio-inspired Network Topology ...	317
<i>Wahiba Taouali, Nicolas Rougier, Frédéric Alexandre</i>	
Use of Swarm Intelligence for the Identification of a Class of Nonlinear Dynamical Systems	331
<i>Syed Z. Rizvi, Hussain N. Al-Duwaish</i>	
Practical Graph Isomorphism for Graphlet Data Mining in Protein Structures	345
<i>Carsten Henneges, Christoph Behle, Andreas Zell</i>	
Learning from Data as an Optimization and Inverse Problem	361
<i>Věra Kůrková</i>	
A Cortically Inspired Learning Model	373
<i>Atif Hashmi, Mikko Lipasti</i>	
Computational Study of Rhythm Propagation Induced by TMS Stimuli in Different Brain Regions	389
<i>Filippo Cona, Melissa Zavaglia, Marcello Massimini, Mario Rosanova, Mauro Ursino</i>	
Smart Growing Cells: Supervising Unsupervised Learning	405
<i>Hendrik Annuth, Christian-A. Bohn</i>	
Author Index	421

Invited Paper

Incremental Kernel Fuzzy c -Means

Timothy C. Havens¹, James C. Bezdek², and Marimuthu Palaniswami²

¹ Michigan State University, East Lansing, MI 48824, U.S.A.

² University of Melbourne, Parkville, Victoria 3010, Australia

havenst@gmail.com, jcbzdek@gmail.com, palani@unimelb.edu.au

Abstract. The size of everyday data sets is outpacing the capability of computational hardware to analyze these data sets. Social networking and mobile computing alone are producing data sets that are growing by terabytes *every day*. Because these data often cannot be loaded into a computer's working memory, most literal algorithms (algorithms that require access to the full data set) cannot be used. One type of pattern recognition and data mining method that is used to analyze databases is clustering; thus, clustering algorithms that can be used on large data sets are important and useful. We focus on a specific type of clustering: kernelized fuzzy c -means (KFCM). The literal KFCM algorithm has a memory requirement of $O(n^2)$, where n is the number of objects in the data set. Thus, even data sets that have nearly 1,000,000 objects require terabytes of working memory—infeasible for most computers. One way to attack this problem is by using incremental algorithms; these algorithms sequentially process chunks or samples of the data, combining the results from each chunk. Here we propose three new incremental KFCM algorithms: rseKFCM, spKFCM, and oKFCM. We assess the performance of these algorithms by, first, comparing their clustering results to that of the literal KFCM and, second, by showing that these algorithms can produce reasonable partitions of large data sets. In summary, the rseKFCM is the most efficient of the three, exhibiting significant speedup at low sampling rates. The oKFCM algorithm seems to produce the most accurate approximation of KFCM, but at a cost of low efficiency. Our recommendation is to use rseKFCM at the highest sample rate allowable for your computational and problem needs.

1 Introduction

The ubiquity of personal computing technology, especially mobile computing, has produced an abundance of staggeringly large data sets—Facebook alone logs over 25 terabytes (TB) of data per day. Hence, there is a great need for algorithms that can address these gigantic data sets. In 1996, Huber [24] classified data set size as in Table 1. Bezdek and Hathaway [17] added the *Very Large* (VL) category to this table in 2006. Interestingly, data with $10^{>12}$ objects is still unloadable on most current (circa 2011) computers. For example, a data set composed of 10^{12} objects, each with 10 features, stored in short integer (4 byte) format would require 40 TB of storage (most high-performance computers have < 1 TB of working memory). Hence, we believe that Table 1 will continue to be pertinent for many years.

Clustering, also called unsupervised learning, numerical taxonomy, typology, and partitioning [41], is an integral part of computational intelligence and machine learning. Often researchers are mired in data sets that are large and unlabeled. There are

Table 1. Huber’s Description of Data Set Sizes [24][17]

Bytes	10^2	10^4	10^6	10^8	10^{10}	10^{12}	$10^{>12}$	∞
“size”	tiny	small	medium	large	huge	monster	VL	infinite

many methods by which researchers can elucidate these data, including projection and statistical methods. Clustering provides another tool for deducing the nature of the data by providing labels that describe how the data separates into groups. These labels can be used to examine the similarity and dissimilarity among and between the grouped objects. Clustering has also been shown to improve the performance of other algorithms or systems by separating the problem-domain into manageable sub-groups—a different algorithm or system is tuned to each cluster [14][6]. Clustering has also been used to infer the properties of unlabeled objects by clustering these objects together with a set of labeled objects (of which the properties are well understood) [29][40].

The problem domains and applications of clustering are innumerable. Virtually every field, including biology, engineering, medicine, finance, mathematics, and the arts, have used clustering. Its function—grouping objects according to context—is a basic part of intelligence and is ubiquitous to the scientific endeavor. There are many algorithms that extract groups from unlabeled object sets: k -means [33][34][35] and c -means [3], and hierarchical clustering [28] being, arguably, the most popular. We will examine a specific, but general, form of clustering: incremental *kernel fuzzy c-means* (KFCM). Specifically, we will develop three new kernel fuzzy clustering algorithms, *random sample and extend* KFCM (rseKFCM), *single-pass* KFCM (spKFCM), and *online* KFCM (oKFCM). The spKFCM and oKFCM algorithms are based on an extension of the *weighted* FCM and weighted kernel k -means models proposed in [22][23][13] and [10], respectively.

1.1 The Clustering Problem

Consider a set of objects $O = \{o_1, \dots, o_n\}$. These objects can represent virtually anything—vintage bass guitars, pure-bred cats, cancer genes expressed in a microarray experiment, cake recipes, or web-pages. The object set O is *unlabeled data*; that is, each object has no associated class label. However, it is assumed that there are subsets of similar objects in O . These subsets are called *clusters*.

Numerical object data is represented as $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^p$, where each dimension of the vector \mathbf{x}_i is a feature value of the associated object o_i . These features can be a veritable cornucopia of numerical descriptions, i.e., RGB values, gene expression, year of manufacture, number of stripes, et cetera.

A wide array of algorithms exists for clustering unlabeled object data O . Descriptions of many of these algorithms can be found in the following general references on clustering: [12][41][3][5][15][44][27][26]. *Clustering* in unlabeled data X is defined as the assignment of *labels* to groups of similar (unlabeled) objects O . In other words, objects are *sorted* or *partitioned* into groups such that each group is composed of objects with similar traits. There are two important factors that all clustering algorithms must consider: 1) the number (and, perhaps, type) of clusters to seek and, 2) a mathematical way to determine the similarity between various objects (or groups of objects).

Let c denote the integer number of clusters. The number of clusters can take the values $c = 1, 2, \dots, n$, where $c = 1$ results in the universal cluster (every object is in one cluster) and $c = n$ results in single-object clusters.

A *partition* of the objects is defined as the set of cn values, where each value $\{u_{ij}\}$ represents the degree to which an object o_i is in (or represented by) the j th cluster. The c -partition is often arrayed as a $n \times c$ matrix $U = [u_{ij}]$, where each column represents a cluster and each row represents an object. There are three types of partitions (to date), crisp, fuzzy (or probabilistic), and possibilistic [3,31] (we do not address possibilistic clustering here).

Crisp partitions of the unlabeled objects are non-empty mutually-disjoint subsets of O such that the union of the subsets cover O . The set of all non-degenerate (no zero columns) crisp c -partition matrices for the object set O is:

$$M_{hcn} = \{U \in \mathbb{R}^{cn} | u_{ij} \in \{0, 1\} \forall i, j; \sum_{j=1}^c u_{ij} = 1 \forall i; \sum_{i=1}^n u_{ij} > 0 \forall j\}, \quad (1)$$

where u_{ij} is the *membership* of object o_i in cluster j ; the partition element $u_{ij} = 1$ if o_i is labeled j and is 0 otherwise.

Fuzzy (or probabilistic) partitions are more flexible than crisp partitions in that each object can have membership in more than one cluster. Note, if U is probabilistic, the partition values are interpreted as the posterior probability $p(j|o_i)$ that o_i is in the j -th class. We assume that fuzzy and probabilistic partitions are essentially equivalent from the point of view of clustering algorithm development. The set of all fuzzy c -partitions is:

$$M_{fcn} = \{U \in \mathbb{R}^{cn} | 0 \leq u_{ij} \leq 1 \forall i, j; \sum_{j=1}^c u_{ij} = 1 \forall i; \sum_{i=1}^n u_{ij} > 0 \forall j\}. \quad (2)$$

Each row of the fuzzy partition U must sum to 1, thus ensuring that every object is completely partitioned ($\sum_i u_{ij} = 1$).

Notice that all crisp partitions are fuzzy partitions, $M_{hcn} \subset M_{fcn}$. Hence, the methods applied here can be easily generalized to kernel HCM.

1.2 FCM

The FCM model is defined as the constrained minimization of

$$J_m(U, V) = \sum_{j=1}^c \sum_{i=1}^n u_{ij}^m \|\mathbf{x}_i - \mathbf{v}_j\|_A^2 \quad (3)$$

where $m \geq 1$ is a fixed fuzzifier and $\|\cdot\|_A$ is any inner product A -induced norm on \mathbb{R}^d , i.e., $\|\mathbf{x}\|_A = \mathbf{x}^T A \mathbf{x}$. Optimal c -partitions U are most popularly sought by using *alternating optimization* (AO) [3,4], but other methods have also been proposed. The *literal* FCM/AO (LFCM/AO) algorithm is outlined in Algorithm 1. There are many ways to initialize LFCM/AO; any method that covers the object space and does not produce identical initial cluster centers would work. We initialize by randomly selecting c feature vectors from the data to serve as initial centers.

Algorithm 1. LFCM/AO to minimize $J_m(U, V)$ [3]

Input: X, c, m **Output:** U, V Initialize V **while** $\max\{\|V_{new} - V_{old}\|^2\} > \epsilon$ **do**

$$u_{ij} = \left[\sum_{k=1}^c \left(\frac{\|\mathbf{x}_j - \mathbf{v}_i\|}{\|\mathbf{x}_j - \mathbf{v}_k\|} \right)^{\frac{2}{m-1}} \right]^{-1}, \quad \forall i, j \quad (4)$$

$$\mathbf{v}_i = \frac{\sum_{j=1}^n (u_{ij})^m \mathbf{x}_j}{\sum_{j=1}^n (u_{ij})^m}, \quad \forall i \quad (5)$$

The alternating steps of LFCM in Eqs. (4) and (5) are iterated until the algorithm terminates, where termination is declared when there are only negligible changes in the cluster center locations: more explicitly, $\max\{\|V - V_{old}\|^2\} < \epsilon$, where ϵ is a pre-determined constant (we use $\epsilon = 10^{-3}$ in our experiments).

It was shown in [2,42,18] that minimizing (3) produces the same result as minimizing the reformulation,

$$J_m(U) = \sum_{j=1}^c \left(\sum_{i=1}^n \sum_{k=1}^n (u_{ij}^m u_{kj}^m d_A^2(\mathbf{x}_i, \mathbf{x}_k)) / 2 \sum_{l=1}^n u_{lj}^m \right), \quad (6)$$

where $d_A^2(\mathbf{x}_i, \mathbf{x}_k) = \|\mathbf{x}_i - \mathbf{x}_k\|_A^2$. This reformulation led to *relational* algorithms, such as RFCM [19] and NERFCM [16], where the data take the relational form $R_A = \|\|\mathbf{x}_i - \mathbf{x}_j\|_A^2\| \in \mathbb{R}^{n \times n}$. Later, we will use (6) to define the KFCM model.

1.3 Related Work on FCM for VL Data

There has been a bevy of research done on clustering in VL data, but only a small portion of this research addresses the fuzzy clustering problem. Algorithms fall in three main categories: i) *Sample and Extend* schemes apply clustering to a (manageably-sized) sample of the full data set, and then non-iteratively extend the sample results to approximate the clustering solution for the remaining data. These algorithms have also been called *extensible* algorithms [36]. An extensible FCM algorithm includes the geFCM [17]. ii) *Incremental* algorithms sequentially load small groups or singletons of the data, clustering each chunk in a single pass, and then combining the results from each chunk. The SPFCM [22] algorithm runs *weighted* FCM (WFCM) on sequential chunks of the data, passing the clustering solution from each chunk onto the next. SPFCM is truly scalable as its space complexity is only based on the size of the sample. A similar algorithm, OFCM [23], performs a similar process as SPFCM; however, rather than passing the clustering solution from one chunk to the next, OFCM clusters the centers from each chunk in one final run. Because of this final run, OFCM is not truly scalable and is not recommended for truly VL data. Another algorithm that is incremental in spirit is brFCM [13], which first bins the data and then clusters the bin centers. However, the efficiency and accuracy results of brFCM are very dependent on the binning

strategy; brFCM has been shown to be very effective on image data, which can be binned very efficiently.

Although not technically an incremental algorithm, but more in the spirit of acceleration, the FFCM algorithm [39] applies FCM to larger and larger nested samples of the data set until there is little change in the solution. Another acceleration algorithm that is incremental in spirit is mrFCM [8], which combines the FFCM with a final literal run of FCM on the full data set. These algorithms are not scalable, however, as they both contain final runs on nearly full-size data set, with one last run on the full data set. iii) *Approximation* algorithms use numerical tricks to approximate the clustering solution using manageable size chunks of the data. Many of these algorithms utilize some sort of data transformation to achieve this goal. The algorithms described in [30,7] fit this description.

None of these algorithms address *kernel* fuzzy clustering, which we describe next.

1.4 KFCM

Consider some non-linear mapping function $\phi : \mathbf{x} \rightarrow \phi(\mathbf{x}) \in \mathbb{R}^{D_K}$, where D_K is the dimensionality of the transformed feature vector \mathbf{x} . Most, if not all, kernel-based methods do not explicitly transform \mathbf{x} and then operate in the higher-dimensional space of $\phi(\mathbf{x})$; instead, they use a kernel function κ that represents the inner product of the transformed feature vectors, $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle$. This kernel function can take many forms, with the polynomial, $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1^T \mathbf{x}_2 + 1)^p$, and *radial-basis-function* (RBF), $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp(\sigma \|\mathbf{x}_1 - \mathbf{x}_2\|^2)$, being two of the most popular forms.

The KFCM algorithm can be generally defined as the constrained minimization of

$$J_m(U; \kappa) = \sum_{j=1}^c \left(\sum_{i=1}^n \sum_{k=1}^n (u_{ij}^m u_{kj}^m d_\kappa(\mathbf{x}_i, \mathbf{x}_k)) / 2 \sum_{l=1}^n u_{lj}^m \right), \quad (7)$$

where $U \in M_{fcn}$, $m > 1$ is the fuzzification parameter, and $d_\kappa(\mathbf{x}_i, \mathbf{x}_k) = \kappa(\mathbf{x}_i, \mathbf{x}_i) + \kappa(\mathbf{x}_k, \mathbf{x}_k) - 2\kappa(\mathbf{x}_i, \mathbf{x}_k)$ is the kernel-based distance between the i th and k th feature vectors.

The KFCM/AO algorithm solves the optimization problem $\min\{J_m(U; \kappa)\}$ by computing iterated updates of

$$u_{ij} = \left(\sum_{k=1}^c \left(\frac{d_\kappa(i, j)}{d_\kappa(i, k)} \right)^{\frac{1}{m-1}} \right)^{-1}, \quad \forall i, j, \quad (8)$$

where, for simplicity, we denote the cluster center to object distance $d_\kappa(\mathbf{x}_i, \mathbf{v}_j)$ as $d_\kappa(i, j)$. This kernel distance is computed as

$$d_\kappa(i, j) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{v}_j)\|^2, \quad (9)$$

where, like LFCM, the cluster centers are linear combinations of the feature vectors,

$$\phi(\mathbf{v}_j) = \frac{\sum_{l=1}^n u_{lj}^m \phi(\mathbf{x}_l)}{\sum_{l=1}^n u_{lj}^m}. \quad (10)$$

Equation (9) cannot be explicitly solved, but by using the identity $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$, denoting $\tilde{\mathbf{u}}_j = \mathbf{u}_j^m / \|\mathbf{u}_j^m\|_1$ (\mathbf{u}_j is the j th column of U), and substituting (10) into (9) we get

$$\begin{aligned}
 d_\kappa(j, i) &= \frac{\sum_{l=1}^n \sum_{s=1}^n u_{lj}^m u_{sj}^m \langle \phi(\mathbf{x}_l), \phi(\mathbf{x}_s) \rangle}{\sum_{l=1}^n u_{lj}^{2m}} \\
 &\quad + \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i) \rangle - 2 \frac{\sum_{l=1}^n u_{lj}^m \langle \phi(\mathbf{x}_l), \phi(\mathbf{x}_i) \rangle}{\sum_{l=1}^n u_{lj}^m} \\
 &= \tilde{\mathbf{u}}_j^T K \tilde{\mathbf{u}}_j + \mathbf{e}_i^T K \mathbf{e}_i - 2 \tilde{\mathbf{u}}_j^T K \mathbf{e}_i \\
 &= \tilde{\mathbf{u}}_j^T K \tilde{\mathbf{u}}_j + K_{ii} - 2(\tilde{\mathbf{u}}_j^T K)_i,
 \end{aligned} \tag{11}$$

where \mathbf{e}_i is the n -length unit vector with the i th element equal to 1. Algorithm 2 outlines the KFCM/AO procedure.

Algorithm 2. KFCM/AO to minimize $J_m(U; \kappa)$

Input: K, c, m

Output: U

Initialize U

while $\max\{\|U_{new} - U_{old}\|^2\} > \epsilon$ **do**

$$d_\kappa(j, i) = \tilde{\mathbf{u}}_j^T K \tilde{\mathbf{u}}_j + K_{ii} - 2(\tilde{\mathbf{u}}_j^T K)_i \quad \forall i, j$$

$$u_{ij} = \left(\sum_{k=1}^c \left(\frac{d_\kappa(i, j)}{d_\kappa(i, k)} \right)^{\frac{1}{m-1}} \right)^{-1} \quad \forall i, j$$

This formulation of KFCM is equivalent to that proposed in [43] and, furthermore, is identical to *relational* FCM (RFCM) [19] if the kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle_A = \mathbf{x}_i^T A \mathbf{x}_j$ is used [20]. If one replaces the relational matrix R in RFCM with $R = -\gamma K$, for any $\gamma > 0$, then RFCM will produce the same partition as KFCM run on K (assuming the same initialization). Likewise, KFCM will produce the same partition as RFCM if $K = -\gamma R$, for any $\gamma > 0$.

1.5 Weighted KFCM

In the KFCM model, each object is considered equally important in the clustering solution. The *weighted* KFCM (wKFCM) model introduces weights that define the relative importance of each object in the clustering solution, similar to the wFCM in [22, 23, 13] and weighted kernel k -means in [10]. The wKFCM model is the constrained minimization of

$$J_{m\mathbf{w}}(U; \kappa) = \sum_{j=1}^c \left(\sum_{i=1}^n \sum_{k=1}^n (w_i w_k u_{ij}^m u_{kj}^m d_\kappa(\mathbf{x}_i, \mathbf{x}_k)) / 2 \sum_{l=1}^n w_l u_{lj}^m \right), \tag{12}$$

where $\mathbf{w} \in \mathbb{R}^n$, $w_i \geq 0 \forall i$, is a set of weights, one element for each feature vector.

The cluster center $\phi(\mathbf{v}_j)$ is a weighted sum of the feature vectors, as shown in (10). Now assume that each object $\phi(\mathbf{x}_i)$ has a different predetermined influence, given by a respective weight w_i . This leads to the definition of the center as

$$\phi(\mathbf{v}_j) = \frac{\sum_{l=1}^n w_l u_{lj}^m \phi(\mathbf{x}_l)}{\sum_{l=1}^n w_l u_{lj}^m}. \quad (13)$$

Substituting (13) into (11) gives

$$\begin{aligned} d_{\kappa}^{\mathbf{w}}(i, j) &= \frac{\sum_{l=1}^n \sum_{r=1}^n w_l w_r u_{lj}^m u_{rj}^m \langle \phi(\mathbf{x}_l), \phi(\mathbf{x}_r) \rangle}{\sum_{l=1}^n w_l^2 u_{lj}^{2m}} \\ &\quad + \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i) \rangle - 2 \frac{\sum_{l=1}^n w_l u_{lj}^m \langle \phi(\mathbf{x}_l), \phi(\mathbf{x}_i) \rangle}{\sum_{l=1}^n w_l u_{lj}^m} \\ &= \frac{1}{\|\mathbf{w} \circ \mathbf{u}_j\|^2} (\mathbf{w} \circ \mathbf{u}_j)^T K (\mathbf{w} \circ \mathbf{u}_j) + K_{ii} \\ &\quad - \frac{2}{\|\mathbf{w} \circ \mathbf{u}_j\|} ((\mathbf{w} \circ \mathbf{u}_j)^T K)_i, \end{aligned} \quad (14)$$

where \mathbf{w} is the vector of predetermined weights and \circ indicates the Hadamard product ($*$ in MATLAB). This leads to the *weighted* KFCM (wKFCM) shown in Algorithm 3. Notice that wKFCM also outputs the index of the nearest object to each cluster center, called the cluster prototype. The vector of indices P is important in the VL data schemes now proposed.

Algorithm 3. wKFCM/AO to minimize $J_{m\mathbf{w}}(U, \kappa)$

Input: K, c, m, \mathbf{w}

Output: U, P

Initialize $U \in M_{fcns}$

while $\max\{\|U_{new} - U_{old}\|^2\} > \epsilon$ **do**

$$d_{\kappa}^{\mathbf{w}}(i, j) = \frac{1}{\|\mathbf{w} \circ \mathbf{u}_j\|^2} (\mathbf{w} \circ \mathbf{u}_j)^T K (\mathbf{w} \circ \mathbf{u}_j) + K_{ii} - \frac{2}{\|\mathbf{w} \circ \mathbf{u}_j\|} ((\mathbf{w} \circ \mathbf{u}_j)^T K)_i \quad \forall i, j$$

$$u_{ij} = \left[\sum_{k=1}^c \left(\frac{d_{\kappa}^{\mathbf{w}}(i, j)}{d_{\kappa}^{\mathbf{w}}(i, k)} \right)^{\frac{1}{m-1}} \right]^{-1} \quad \forall i, j$$

$$p_j = \arg \min_i \{d_{\kappa}(i, j)\}, \quad \forall j$$

2 Incremental Algorithms

The problem with KFCM and wKFCM is that they require working memory to store the full $n \times n$ kernel matrix K . For large n , the memory requirement can be very significant; e.g., with $n = 1,000,000$, 4 TB of working memory are required. This is infeasible for even most high-performance computers. Hence, even Huber's "medium" data sets (on the order of 10^6) are impossible to cluster on moderately powerful computers. For this reason, kernel clustering of large-scale data is infeasible without some method that scales well.

2.1 rseKFCM

The most basic, and perhaps obvious, way to address kernel fuzzy clustering in VL data is to sample the dataset and then use KFCM to compute partitions of the sampled data. This is similar to the approach of geFFCM (geFFCM uses literal FCM instead of KFCM); however, geFFCM uses a progressive sampling¹ approach to draw a sample that is representative (enough) of the full data set. However, for VL data, this representative sample may be large itself. Thus, we use randomly draw without replacement a predetermined sub-sample of X . We believe that random sampling is sufficient for VL data and is, of course, computationally less expensive than a progressive approach. There are other sampling schemes addressed in [11,32,11]; these papers also support our claim that uniform random sampling is preferred. Another issue with directly applying the sample and extend approach of geFFCM is that it first computes cluster centers V and then uses (4) to extend the partition to the remaining data. In contrast, KFCM does not return a cluster center (per se); hence, one cannot directly use (4) to extend the partition to the remaining data. However, recall that wKFCM, in Algorithm 3, returns a set of cluster prototypes P , which are the indices of the c objects that are closest to the cluster centers (in the RKHS). Thus follows our rseKFCM algorithm, outlined in Algorithm 4.

Algorithm 4. rseKFCM to approximately minimize $J_m(U; \kappa)$

Input: Kernel function κ , X , c , m , n_s

Output: U

- 1 Sample n_s vectors from X , denoted X_s
- 2 $K = [\kappa(\mathbf{x}_i, \mathbf{x}_j)]$, $\forall \mathbf{x}_i, \mathbf{x}_j \in X_s$
- 3 $U, P = \text{wKFCM}(K, c, m, \mathbf{1}_{n_s})$
- 4 Extend partition to X :

$$d_\kappa(j, i) = \kappa(\mathbf{x}_i, \mathbf{x}_i) + \kappa(\mathbf{x}_{P_j}, \mathbf{x}_{P_j}) - 2\kappa(\mathbf{x}_i, \mathbf{x}_{P_j}) \quad \forall i, j$$

$$u_{ij} = \left[\sum_{k=1}^c \left(\frac{d_\kappa(i, j)}{d_\kappa(i, k)} \right)^{\frac{1}{m-1}} \right]^{-1} \quad \forall i, j$$

First, a random sample of X is drawn at Line 1. Then the kernel matrix K is computed for this random sample at Line 2, and at Line 3 wKFCM is used to produce a set of cluster prototypes. Finally, the partition is extended to the remaining data at Line 4.

The rseKFCM algorithm is scalable as one can choose a sample size n_s to suit their computational resources. However, the clustering iterations are not performed on the entire data set (in literal or in chunks); hence, if the sample X_s is not representative of the full data set, then rseKFCM will not accurately approximate the literal KFCM solution.

This leads to the discussion of two algorithms that operate on the full data set by separating it into multiple chunks.

¹ [37] provide a very readable analysis and summary of progressive sampling schemes.

2.2 spKFCM

The spKFCM algorithm is based upon the spFCM algorithm proposed in [22]. Essentially, spKFCM (and spFCM) splits the data into multiple (approximately) equally size chunks, then clusters each chunk separately. The clustering result from each chunk is passed to the next chunk in order to approximate the literal KFCM solution on the full data set. In SPFCM, the cluster center locations from each chunk are passed to the next chunk as data points to be included in the data set that is clustered. However, these cluster centers are weighted in the WFCM by the sum of the respective memberships, i.e. the c th cluster is weighted by the sum of the c th row of U . Essentially, the weight causes the cluster centers to have more influence on the clustering solution that the data in the data chunk. In other words, each cluster center represents the multiple data points in each cluster.

Because there are no cluster centers in KFCM (or wKFCM), the data that is passed on to the next chunk are, instead, the cluster prototypes—the objects nearest to the cluster center in the RKHS. Hence, the kernel matrix for each data chunk is the $(n_s + c) \times (n_s + c)$ kernel function results— n_s columns (or rows) for the objects in the data chunk and c columns for the c prototypes passed on from the previous chunk.

Algorithm 5. spKFCM to approximately minimize $J_m(U; \kappa)$

Input: Kernel function κ , X , c , m , s

Output: P

- 1 Randomly draw s (approximately) equal-sized subsets of the integers $\{1, \dots, n\}$, denoted $\Xi = \{\xi_1, \dots, \xi_s\}$. n_l is the cardinality of ξ_l .
 - 2 $K = [\kappa(\mathbf{x}_i, \mathbf{x}_j)] \quad i, j = \xi_1$
 - 3 $U, P = \text{wKFCM}(K, c, m, \mathbf{1}_{n_1})$
 - 4 $w'_j = \sum_{i=1}^{n_1} u_{ij} \quad \forall j$
 - for** $l = 2$ **to** s **do**
 - 5 $\mathbf{w} = \{w', \mathbf{1}_{n_l}\}$
 - 6 $\xi' = \{\xi'(P), \xi_l\}$
 - 7 $K = [\kappa(\mathbf{x}_i, \mathbf{x}_j)] \quad i, j = \xi'$
 - 8 $U, P = \text{wKFCM}(K, c, m, \mathbf{w})$
 - 9 $w'_j = \sum_{i=1}^{n_l+c} u_{ij} \quad \forall j$
 - 10 $P = \xi'(P)$
-

Algorithm 5 outlines the spKFCM algorithm. At Line 1, the data X is randomly separated into s equally-sized subsets, where the indices of the objects in each subset of denoted ξ_i , $i = 1, \dots, s$. Lines 2-4 comprise the operations on the first data chunk. At Line 2, the $n_s \times n_s$ kernel matrix of the first data chunk is computed and, at Line 3, wKFCM is used to cluster the first data chunk. The weights of the c cluster prototypes returned by wKFCM are computed at Line 4. Lines 5-9 are the main loop of spKFCM. For each data chunk, Line 5 creates a vector of weights, where the weight for the c prototypes is calculated from the previous data chunk results and the weights of the n_s objects are set to 1. At Line 6, the indices of the objects in the l th data chunk and the c cluster prototypes are combined and, at Line 7, the $(n_s + c) \times (n_s + c)$ kernel matrix

is computed (for the objects indexed by ξ_l and the c cluster prototypes). wKFCM is then used to produce the clustering results at Line 8. And, at Line 9, the weights are calculated, which are then used in the next data chunk loop. Finally, at Line 10, the indices of the c cluster prototypes are returned.

spKFCM is a scalable algorithm because one can choose the size of the data chunk to be clustered and the maximum size of the data to be clustered is $n_s + c$. The storage requirements for the kernel matrices is thus $O((n_s + c)^2)$.

2.3 oKFCM

The oKFCM algorithm is similar to spKFCM, and is based on the oFCM algorithm proposed in [23]. Algorithm 6 outlines the oKFCM procedure. Like spKFCM, oKFCM starts by separating the objects into s equally-sized data chunks. However, unlike spKFCM, it does not pass the cluster prototypes from the previous data chunk onto the next data chunk. oKFCM simply calculates s sets of c cluster prototypes, one set from each data chunk. It then computes a weight for each of the cs cluster prototypes, which is the sum of the row of the respective membership matrix (there is one membership matrix computed for each of the s data chunks). Finally, the cs cluster prototypes are partitioned using wKFCM, producing a final set of c cluster prototypes.

Algorithm 6. oKFCM to approximately minimize $J_m(U; \kappa)$

Input: Kernel function κ , X , c , m , s

Output: U, P

- 1 Randomly draw s (approximately) equal-sized subsets of the integers $\{1, \dots, n\}$, denoted $\Xi = \{\xi_1, \dots, \xi_s\}$. n_l is the cardinality of ξ_l .
 - 2 $K = [\kappa(\mathbf{x}_i, \mathbf{x}_j)] \quad i, j = \xi_1$
 - 3 $U_1, P_1 = \text{wKFCM}(K, c, m, \mathbf{1}_{n_1})$
 - for** $l = 2$ **to** s **do**
 - 4 $K = [\kappa(\mathbf{x}_i, \mathbf{x}_j)] \quad i, j = \xi_l$
 - 5 $U_l, P'_l = \text{wKFCM}(K, c, m, \mathbf{1}_{n_l})$
 - 6 $P_l = \xi_l(P'_l)$
 - 7 $P_{all} = \{P_1, \dots, P_s\}$
 - 8 $K = [\kappa(\mathbf{x}_i, \mathbf{x}_j)] \quad i, j = P_{all}$
 - 9 $\mathbf{w}_l = \sum_{j=1}^{n_s} (U_l)_{.j} \quad \forall l$
 - 10 $U, P' = \text{wKFCM}(K, c, m, \mathbf{w})$
 - 11 $P = P_{all}(P')$
-

Because there is no interaction between the initial clustering done on each data chunk, oKFCM could be easily implemented on a distributed architecture. Each iteration of Lines 4-6 is independent and could thus be simultaneously computed on separate nodes of a parallel architecture (or cloud, if you will). However, the final clustering of the cs cluster prototypes prevents oKFCM from being a truly scalable algorithm. If s is large, then this final data set is large. In extreme cases, if $s \gg n_s$ then the storage requirement for oKFCM becomes $O((cs)^2)$.

3 Experiments

We performed two sets of experiments. The first compared the performance of the incremental KFCM algorithms on data for which there exists ground-truth (or known object labels). The second set of experiments applies the proposed algorithms to data sets for which there exists no ground-truth. For these data, we compared the partitions from the incremental KFCM algorithms to the literal KFCM partitions.

For all algorithms, we initialize U by randomly choosing c objects as the initial cluster centers. The value of $\epsilon = 10^{-3}$ and the fuzzifier $m = 1.7$. The termination criteria is $\max\{\|U_{new} - U_{old}\|^2\} < \epsilon$. The experiments were performed on a single core of an AMD Opteron in a Sun Fire X4600 M2 server with 256 gigabytes of memory. All code was written in the MATLAB computing environment.

3.1 Evaluation Criteria

We judge the performance of the incremental KFCM algorithms using three criteria. Each criteria is computed for 21 independent runs with random initializations and samplings. The results presented are the average values.

1. **Speedup Factor or Run-time.** This criteria represents an actual run-time comparison. When the KFCM solution is available, speedup is defined as $t_{literal}/t_{incremental}$, where these values are times in seconds for computing the membership matrix U . When the data is too large to compute KFCM solutions, we present run-time in seconds for the incremental algorithms.
2. **Adjusted Rand Index.** The Rand index [38] is a measure of agreement between two crisp partitions of a set of objects. One of the two partitions is usually a reference partition U' , which represents the ground truth labels for the objects in the data. In this case the value $R(U, U')$ measures the degree to which a candidate partition U matches U' . A Rand index of 1 indicates perfect agreement, while a Rand index of 0 indicates perfect disagreement. The version that we use here, the *adjusted Rand index*, $ARI(U, U')$, is a bias-adjusted formulation developed by Hubert and Arabie [25]. To compute the ARI, we first harden the fuzzy partitions by setting the maximum element in each row of U to 1, and all else to 0. We use the ARI to compare the clustering solutions to ground-truth labels (when available), and also to compare the VL data algorithms to the literal FCM solutions.

Note that the rseKFCM, spKFCM, and oKFCM algorithms do not produce full data partitions; they produce cluster prototypes as output. Hence, we cannot directly compute ARI and fuzzy ARI for these algorithms. To complete the calculations, we used the Extension step to produce full data partitions from the output cluster prototypes. The Extension step was *not* included in the speedup factor or run-time calculations for these algorithms as these algorithms were designed to return cluster prototypes (as the analogous rseFCM, SPFCM, and OFCM), not full data partitions. However, we observed in our experiments that the Extension step added a nearly negligible amount of time to the overall run-time of the algorithms.

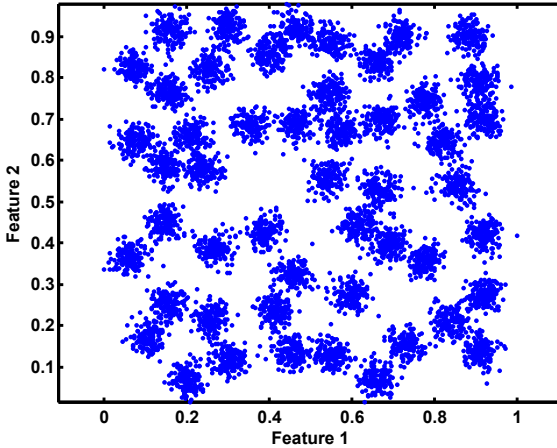


Fig. 1. 2D50 synthetic data; $n = 7500$ objects, $c = 50$ clusters

The data we used in this study are:

1. **2D50** ($n = 7,500$, $c = 50$, $d = 2$). These data are composed of 7,500 2-dimensional vectors, with a visually-preferred grouping into 50 clusters [2]. Figure 1 shows a plot of these data. An RBF kernel with $\sigma = 1$, $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(\sigma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, was used.
2. **MNIST** ($n = 70,000$, $c = 10$, $d = 784$). This data set is a subset of the collection of handwritten digits available from the *National Institute of Standards and Technology* (NIST) [3]. There are 70,000 28×28 pixel images of the digits 0 to 9. Each pixel has an integer value between 0 and 255. We normalize the pixel values to the interval $[0, 1]$ by dividing by 255 and concatenate each image into a 784-dimensional column vector. A 5-degree inhomogeneous polynomial kernel was used, which was shown to be (somewhat) effective in [21][9][45].

Figure 2 shows the results of the incremental algorithms on the 2D50 data set. The speedup factor, shown in view (a), demonstrates that rseKFCM is the fastest algorithm overall, with a speedup of about 450 at a 1% sample rate. However, at sample rates $> 5\%$, rseKFCM and spKFCM exhibit nearly equal speedup results. As view (b) shows, at sample rates $> 5\%$, all three algorithms perform comparably. The rseKFCM algorithm shows slightly better results than oKFCM at sample rates $> 5\%$ and the spKFCM algorithm exhibits inconsistent performance, sometimes performing better than the other algorithms, sometimes worse; although, all three algorithms show about the same performance. The oKFCM shows the best performance at very low sample rates ($< 5\%$) but the oKFCM algorithm is also the least efficient of the three.

Figure 3 shows the performance of the incremental KFCM algorithms on the MNIST data set. These results tell a somewhat different story than the previous data set. First,

² The 2D50 data were designed by Ilja Sidoroff and can be downloaded at <http://cs.joensuu.fi/~isido/clustering/>

³ The MNIST data can be downloaded at <http://yann.lecun.com/exdb/mnist/>

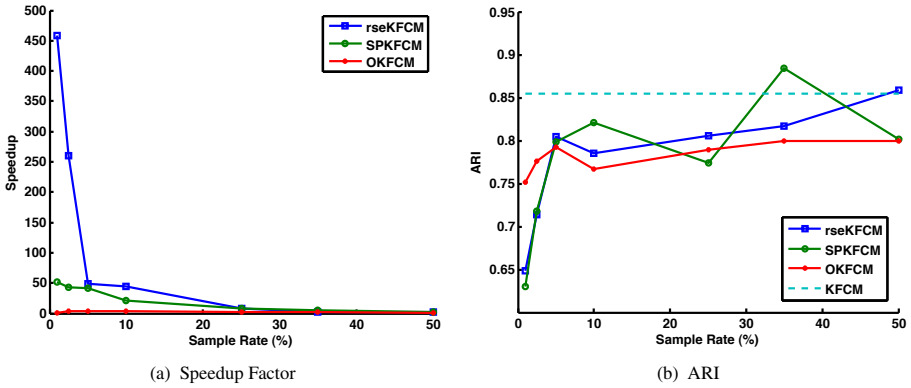


Fig. 2. Performance of incremental KFCM algorithms on 2D50 data set. ARI is calculated relative to ground-truth labels.

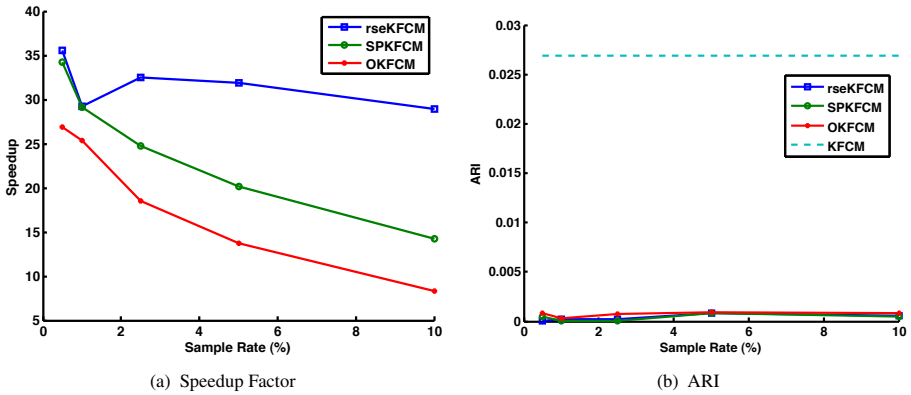


Fig. 3. Performance of incremental KFCM algorithms on MNIST data set. ARI is calculated relative to ground-truth labels.

rseKFCM is no longer the clear winner in terms of speedup. At low sample rates, the spKFCM and rseKFCM perform comparably. This is because rseKFCM suffered from slow convergence at the low sample rates. At sample rates $> 2\%$, rseKFCM is clearly the fastest algorithm. Second, oKFCM is comparable to the other algorithms in terms of speedup. However, the ARI results show a dismal view of the performance of these algorithms for this data set (in terms of accuracy compared to ground truth labels). All three algorithms fail to perform nearly as well as the literal KFCM algorithm (shown by the dotted line). Note that even the literal KFCM performs rather poorly on this data set (in terms of its accuracy with respect to comparison with the ground truth labels). This suggests that the incremental KFCM algorithms have trouble with data sets that are difficult to cluster. Perhaps, the cluster structure is lost when the kernel matrix is sampled.

4 Discussion and Conclusions

We present here three adaptations of an incremental FCM algorithm to kernel FCM. In a nutshell, the rseKFCM algorithm seems to be the preferred algorithm. It is the most scalable and efficient solution, and produces results that are on par with those of spKFCM and oKFCM. The oKFCM does not suffer in performance at low sample rates, but is also the most inefficient of the three. Hence, we recommend using rseKFCM at the highest sample rate possible for your computational resources. We believe that this approach will yield the most desirable results, in terms of both in speedup and accuracy.

Although the incremental KFCM algorithms performed well on the synthetic 2D50 data set, their performance suffered, relative to literal KFCM, on the MNIST data set, which was not as “easily” clustered. To combat this issue, we are going to examine other ways by which the KFCM solution can be adapted for incremental operation. One method we are currently examining is a way by which a more meaningful cluster prototype can be produced by wKFCM. Furthermore, we plan to look at ways that the sample size can be increased without sacrificing speedup and scalability, such as in the approximate kernel k -means approach proposed in [921].

Another question that arises in incremental clustering is validity or, in other words, the quality of the clustering. Many cluster validity measures require full access to the objects’ vector data or to the full kernel (or relational) matrix. Hence, we aim to extend several well-known cluster validity measures for incremental use by using similar strategies to the adaptations presented here.

In closing, we would like emphasize that clustering algorithms, by design, are meant to find the natural groupings in *unlabeled* data (or to discover unknown trends in labeled data). Thus, the effectiveness of a clustering algorithm cannot be appropriately judged by pretending it is a classifier and presenting classification results on labeled data, where each cluster is considered to be a class label. Although we did compare against ground-truth labels in this paper, we used these experiments to show how well the incremental KFCM schemes were successful in producing similar partitions to those produced by literal FCM, which was our bellwether of performance. This will continue to be our standard for the work ahead.

Acknowledgements. Timothy Havens is supported by the National Science Foundation under Grant #1019343 to the Computing Research Association for the CI Fellows Project.

We wish to acknowledge the support of the Michigan State University High Performance Computing Center and the Institute for Cyber Enabled Research.

References

1. Belabbas, M., Wolfe, P.: Spectral methods in machine learning and new strategies for very large datasets. *Proc. National Academy of Sciences* 106(2), 369–374 (2009)
2. Bezdek, J.: A convergence theorem for the fuzzy isodata clustering algorithms. *IEEE Trans. Pattern Analysis and Machine Intelligence* 2, 1–8 (1980)
3. Bezdek, J.: *Pattern Recognition With Fuzzy Objective Function Algorithms*. Plenum, New York (1981)

4. Bezdek, J., Hathaway, R.: Convergence of alternating optimization. *Nueral, Parallel, and Scientific Computations* 11(4), 351–368 (2003)
5. Bezdek, J., Keller, J., Krishnapuram, R., Pal, N.: *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Kluwer, Norwell (1999)
6. Bo, W., Nevatia, R.: Cluster boosted tree classifier for multi-view, multi-pose object detection. In: *Proc. ICCV* (October 2007)
7. Cannon, R., Dave, J., Bezdek, J.: Efficient implementation of the fuzzy c -means algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence* 8, 248–255 (1986)
8. Cheng, T., Goldgof, D., Hall, L.: Fast clustering with application to fuzzy rule generation. In: *Proc. IEEE Int. Conf. Fuzzy Systems*, Tokyo, Japan, pp. 2289–2295 (1995)
9. Chitta, R., Jin, R., Havens, T., Jain, A.: Approximate kernel k -means: Solution to large scale kernel clustering. In: *Proc. ACM SIGKDD* (2011)
10. Dhillon, I., Guan, Y., Kulis, B.: Kernel k -means, spectral clustering, and normalized cuts. In: *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery Data Mining*, pp. 551–556 (August 2004)
11. Drineas, P., Mahoney, M.: On the nystrom method for approximating a gram matrix for improved kernel-based learning. *The J. of Machine Learning Research* 6, 2153–2175 (2005)
12. Duda, R., Hart, P., Stork, D.: *Pattern Classification*, 2nd edn. Wiley-Interscience (October 2000)
13. Eschrich, S., Ke, J., Hall, L., Goldgof, D.: Fast accurate fuzzy clustering through data reduction. *IEEE Trans. Fuzzy Systems* 11, 262–269 (2003)
14. Frigui, H.: Simultaneous Clustering and Feature Discrimination with Applications. In: *Advances in Fuzzy Clustering and Feature Discrimination with Applications*, pp. 285–312. John Wiley and Sons (2007)
15. Hartigan, J.: *Clustering Algorithms*. Wiley, New York (1975)
16. Hathaway, R., Bezdek, J.: NERF c -MEANS: Non-euclidean relational fuzzy clustering. *Pattern Recognition* 27, 429–437 (1994)
17. Hathaway, R., Bezdek, J.: Extending fuzzy and probabilistic clustering to very large data sets. *Computational Statistics and Data Analysis* 51, 215–234 (2006)
18. Hathaway, R., Bezdek, J., Tucker, W.: An improved convergence theory for the fuzzy iso-data clustering algorithms. In: Bezdek, J. (ed.) *Analysis of Fuzzy Information*, vol. 3, pp. 123–132. CRC Press, Boca Raton (1987)
19. Hathaway, R., Davenport, J., Bezdek, J.: Relational duals of the c -means clustering algorithms. *Pattern Recognition* 22(2), 205–212 (1989)
20. Hathaway, R., Huband, J., Bezdek, J.: A kernelized non-euclidean relational fuzzy c -means algorithm. In: *Proc. IEEE Int. Conf. Fuzzy Systems*, pp. 414–419 (2005)
21. Havens, T., Chitta, R., Jain, A., Jin, R.: Speedup of fuzzy and possibilistic c -means for large-scale clustering. In: *Proc. IEEE Int. Conf. Fuzzy Systems*, Taipei, Taiwan (2011)
22. Hore, P., Hall, L., Goldgof, D.: Single pass fuzzy c means. In: *Proc. IEEE Int. Conf. Fuzzy Systems*, London, England, pp. 1–7 (2007)
23. Hore, P., Hall, L., Goldgof, D., Gu, Y., Maudsley, A.: A scalable framework for segmenting magnetic resonance images. *J. Signal Process. Syst.* 54(1-3), 183–203 (2009)
24. Huber, P.: Massive Data Sets Workshop: The Morning After. In: *Massive Data Sets*, pp. 169–184. National Academy Press (1997)
25. Hubert, L., Arabie, P.: Comparing partitions. *J. Classification* 2, 193–218 (1985)
26. Jain, A., Dubes, R.: *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs (1988)
27. Jain, A., Murty, M., Flynn, P.: Data clustering: A review. *ACM Computing Surveys* 31(3), 264–323 (1999)
28. Johnson, S.: Hierarchical clustering schemes. *Psychometrika* 2, 241–254 (1967)
29. Khan, S., Situ, G., Decker, K., Schmidt, C.: Go Figure: Automated Gene Ontology annotation. *Bioinf.* 19(18), 2484–2485 (2003)

30. Kolen, J., Hutcheson, T.: Reducing the time complexity of the fuzzy c-means algorithm. *IEEE Trans. Fuzzy Systems* 10, 263–267 (2002)
31. Krishnapuram, R., Keller, J.: A possibilistic approach to clustering. *IEEE Trans. on Fuzzy Sys.* 1(2) (May 1993)
32. Kumar, S., Mohri, M., Talwalkar, A.: Sampling techniques for the nystrom method. In: *Proc. Conf. Artificial Intelligence and Statistics*, pp. 304–311 (2009)
33. Lloyd, S.: Least square quantization in pcm. Tech. rep., Bell Telephone Laboratories (1957)
34. Lloyd, S.: Least square quantization in pcm. *IEEE Trans. Information Theory* 28(2), 129–137 (1982)
35. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: *Proc. 5th Berkeley Symp. Math. Stat. and Prob.*, pp. 281–297. University of California Press (1967)
36. Pal, N., Bezdek, J.: Complexity reduction for “large image” processing. *IEEE Trans. Systems, Man, and Cybernetics B* (32), 598–611 (2002)
37. Provost, F., Jensen, D., Oates, T.: Efficient progressive sampling. In: *Proc. KDDM*, pp. 23–32 (1999)
38. Rand, W.: Objective criteria for the evaluation of clustering methods. *J. Amer. Stat. Assoc.* 66(336), 846–850 (1971)
39. Shankar, B.U., Pal, N.: FFCM: an effective approach for large data sets. In: *Proc. Int. Conf. Fuzzy Logic, Neural Nets, and Soft Computing*, Fukuoka, Japan, p. 332 (1994)
40. The UniProt Consortium: The universal protein resource (UniProt). *Nucleic Acids Res.* 35, D193–D197 (2007)
41. Theodoridis, S., Koutroumbas, K.: *Pattern Recognition*, 4th edn. Academic Press, San Diego (2009)
42. Tucker, W.: Counterexamples to the convergence theorem for fuzzy isodata clustering algorithms. In: Bezdek, J. (ed.) *Analysis of Fuzzy Information*, vol. 3, pp. 109–122. CRC Press, Boca Raton (1987)
43. Wu, Z., Xie, W., Yu, J.: Fuzzy c-means clustering algorithm based on kernel method. In: *Proc. Int. Conf. Computational Intelligence and Multimedia Applications*, pp. 49–54 (September 2003)
44. Xu, R., Wunsch II, D.: *Clustering*. IEEE Press, Piscataway (2009)
45. Zhang, R., Rudnicky, A.: A large scale clustering scheme for kernel k-means. In: *Proc. Int. Conf. Pattern Recognition*, pp. 289–292 (2002)

Part I
Evolutionary Computation

Ant Algorithm for Optimal Sensor Deployment

Stefka Fidanova¹, Pencho Marinov¹, and Enrique Alba²

¹ Institute of Information and Communication Technologies, Bulgarian Academy of Sciences
Acad. G. Bonchev str. bl.25A, 1113 Sofia, Bulgaria
{stefka,pencho}@parallel.bas.bg

² E.T.S.I. Informática, Grupo GISUM (NEO), University of Malaga, Malaga, Spain
eat@lcc.uma.es

Abstract. Telecommunications is a general term for a vast array of technologies that send information over distances. Mobile phones, land lines, satellite phones and voice over Internet protocol are all telephony technologies - just one field of telecommunications. Radio, television and networks are a few more examples of telecommunication. Nowadays, the trend in telecommunication networks is having highly decentralized, multi-node networks. From small, geographically close, size-limited local area networks the evolution has led to the huge worldwide Internet. In this context Wireless Sensor Networks (WSN) have recently become a hot topic in research. When deploying a WSN, the positioning of the sensor nodes becomes one of the major concerns. One of the objectives is to achieve full coverage of the terrain (sensor field). Another objectives are also to use a minimum number of sensor nodes and to keep the connectivity of the network. In this paper we address a WSN deployment problem in which full coverage and connectivity are treated as constraints, while objective function is the number of the sensors. To solve it we propose Ant Colony Optimization (ACO) algorithm.

1 Introduction

Telecommunications are an important symbol of our present information society. Telecommunication is a field in which many open research lines are challenging the research community. Nowadays, the trend in telecommunication networks is having highly decentralized, multi-node networks. From small, geographically close, size-limited local area networks the evolution has led to the huge worldwide Internet. This same path is followed by wireless communications, where we can already see wireless telephony reaching virtually any city in the world. Wireless Sensor Networks (WSN) allow the monitoring of wide and remote areas with precision and liveness unseen to the date without the intervention of a human operator. The evolution of wireless networking technologies and their key role in Future Internet scenarios offers an increasing wealth of opportunities for distributing data over wireless networks. A WSN allows an administrator to automatically and remotely monitor almost any phenomenon with a high precision. The use of multiple small cooperative devices yields a brand new horizon of possibilities yet offers a great amount of new problems to be solved. WSN have so far been employed in military activities such as reconnaissance, surveillance, and target acquisition [5], environmental activities such as forest fire prevention, geophysical

activities such as volcano eruptions study [13], biomedical purposes such as health data monitoring [14] or civil engineering [11].

The wireless sensors, fulfill two fundamental functions: sensing and communicating. The sensing can be of different types (seismic, acoustic, chemical, optical, etc.), and the communication is performed wirelessly. However, the small size and energy storage capacity of the sensors prevent them from relaying their gathered information directly to the base. It is therefore necessary that they transmit their data to a high energy communication node (HECN) able to provide the transmission relay to an aircraft or a satellite. All sensors must be able to transmit their data to this node, either directly or via hops, using nearby sensors as communication relays.

When deploying a WSN, the positioning of the sensor nodes becomes one of the major concerns. The coverage obtained with the network and the economic cost of the network depend directly of it. Since many WSN can have large numbers of nodes, the task of selecting the geographical positions of the nodes for an optimally designed network can be very complex. Therefore, metaheuristics seem an interesting option to solve this problem.

In this paper we propose a solution method for the WSN layout problem using ACO. We focus on minimizing the number of nodes, while the full coverage of the network and connectivity are considered as constraints.

Jourdan [7] solved an instance of WSN layout using a multiobjective genetic algorithm. In there formulation a fixed number of sensors had to be placed in order to maximize the coverage. In [9] are proposed several evolutionary algorithms to solve the problem.

The rest of the paper is organized as follows. In Section 2 the WSN is described and the deployment problem is formulated. Section 3 presents the ACO algorithm. The existing state of the art is briefly reviewed in Section 4. In Section 5 the experimental results obtained are shown. Finally, several conclusions are drawn in Section 6.

2 Problem Formulation

Mobile communications is a major area in the industry. Customers get used to having mobility and connectivity, thus this types of services are required more and more. Mobile communications require the use of a mobile device, the presence of a network accessible and a network that manage the connections and communications. Also, ad hoc and sensor networks need to define a cluster responsible for communications to take place. The most important for companies is to can offer best possible services at the lowest cost.

A Wireless Sensor Network is a wireless network formed by sensor nodes. Each sensor node sens an area around itself called its sensing area. A parameter called sensing radius determines the sensitivity range of the sensor node and thus the sensing area. The nodes communicate among themselves using wireless communication links. These links are determined by a communication radius. A special node in the WSN called High Energy Communication Node (HECN) is responsible for external access to the network. Therefore, every sensor node in the network must have communication with the HECN. Since the communication radius is often much smaller than the network size,

direct links are not possible for peripheral nodes. A multi-hop communication path is then established for those nodes that do not have the HECN within their communication range.

The WSN layout problem amounts to deciding the geographical position of the sensor nodes that form a WSN. In our formulation, a non-fixed amount of sensor nodes has to be placed in a terrain providing full sensitivity coverage. The positions of the nodes have to be chosen in a way that minimizes the total number of sensor nodes, while keeps the connectivity of the network.

The WSN operates by rounds: In a round, every node collects the data from its measurements and sends it to the HECN. Every node transmits the information packets to the neighbor that is closest to the HECN, or the HECN itself if it is within the communication range. The sensing area of the WSN is the union of the individual areas of all nodes. The designer wants the network to cover the complete sensing area. On the other hand, the number of sensor nodes must be kept as low as possible, since using many nodes represents a high cost of the network, possibly influences of the environment and also provokes a probability of detection (when stealth monitoring is designed). The objective of this problem is to minimize the number of sensors deployed while the area is fully covered and connected.

3 Ant Colony Optimization Framework

Many of the existing solutions to this problem come from the field of Evolutionary Computation [19]. After analyzing them, we noticed that these interesting developments are quite similar to ACO algorithms. The relation between ACO algorithms and evolutionary algorithms provides a structural way of handling constrained problems. They have in common the use of a probabilistic mechanisms for recombination of individuals. This leads to algorithms where the population statistics are kept in a probability vector. In each iteration of the algorithm, these probabilities are used to generate new solutions. The new solutions are then used to adapt the probability vector.

Real ants foraging for food lay down quantities of pheromone (chemical cues) marking the path that they follow. An isolated ant moves essentially guided by an heuristic function and an ant encountering a previously laid pheromone will detect and decide to follow it with high probability thus taking more informed actions based on the experience of previous ants (and thereby reinforce it with a further quantity of pheromone). The repetition of the above mechanism represents the auto-catalytic behavior of real ant colony where the more the ants follow a trail, the more attractive that trail becomes.

The ACO algorithm uses a colony of artificial ants that behave as cooperative agents in a mathematic space were they are allowed to search and reinforce pathways (solutions) in order to find the optimal ones. The problem is represented by graph and the ants walk on the graph to construct solutions. The solution is represented by a path in the graph. After initialization of the pheromone trails, ants construct feasible solutions, starting from random nodes, then the pheromone trails are updated. At each step ants compute a set of feasible moves and select the best one (according to some probabilistic rules based on a heuristic guided function) to carry out the rest of the tour. The structure of ACO algorithm is shown in Figure 1. The transition probability p_{ij} , to chose the

node j when the current node is i , is based on the heuristic information η_{ij} and on the pheromone trail level τ_{ij} of the move, where $i, j = 1, \dots, n$.

$$p_{ij} = \frac{\tau_{ij}^\alpha \eta_{ij}^\beta}{\sum_{k \in allowed} \tau_{ik}^\alpha \eta_{ik}^\beta} \quad (1)$$

The higher value of the pheromone and the heuristic information, the more profitable is to select this move. In the beginning, the initial pheromone level is set to a small positive constant value τ_0 and then ants update this value after completing the construction stage [3]. ACO algorithms adopt different criteria to update the pheromone level.

Ant Colony Optimization

```

Initialize number of ants;
Initialize the ACO parameters;
while not end-condition do
    for k=0 to number of ants
        ant k starts from a random node;
        while solution is not constructed do
            ant k selects higher probability node;
        end while
    end for
    Local search procedure;
    Update-pheromone-trails;
end while

```

Fig. 1. Pseudocode for ACO

In our implementation we use MAX-MIN Ant System (MMAS) [12], which is one of the more popular ant approaches. The main feature of MMAS is using a fixed upper bound τ_{max} and a lower bound τ_{min} of the pheromone trails. Thus the accumulation of big amounts of pheromone by part of the possible movements and repetition of same solutions is partially prevented. The main features of MMAS are:

- Strong exploration to the space search of the best found solution. This can be achieved by only allowing one single ant to add pheromone after each iteration (the best one).
- Wide exploration of the best solution. After the first iteration, the pheromone trails are reinitialized to τ_{max} . In the next iteration, only the movements that belong to the best solution receive a pheromone, while the rest pheromone values are only evaporated.

The aim of using only one solution is to make the solution components, which frequently occur in the best found solutions, get a larger reinforcement. The pheromone trail update rule is given by:

$$\tau_{ij} \leftarrow \rho \tau_{ij} + \Delta \tau_{ij}, \quad (2)$$

$$\Delta\tau_{ij} = \begin{cases} 1/C(V_{best}) & \text{if } (i, j) \in \text{best solution} \\ 0 & \text{otherwise} \end{cases},$$

Where V_{best} is the iteration best solution and $i, j = 1, \dots, n$, $\rho \in [0, 1]$ models evaporation in the nature. To avoid stagnation of the search, the range of possible pheromone values on each movement is limited to an interval $[\tau_{min}, \tau_{max}]$. τ_{max} is an asymptotic maximum of τ_{ij} and $\tau_{max} = 1/(1 - \rho)C(V^*)$, while $\tau_{min} = 0.087\tau_{max}$. Where V^* is the optimal solution, but it is unknown, therefore we use V_{best} instead of V^* .

The WSN layout problem is represented by graph as follows: the terrain is modeled by grid $G = \{g_{ij}\}_{N \times M}$; the pheromone is related with location sites $Ph = \{ph_{ij}\}_{N \times M}$, the initial pheromone can be a small value, for example $1/n_{ants}$. The central point, where the HECN is located, is included in the solutions like first point (zero point). Every ant starts to create the rest of the solution from a random node which communicates with central one, thus the different start of every ant in every iteration is guaranteed. The ant chooses the next position by the ACO probabilistic rule (equation 1). It chooses the point having the higher probability.

The following heuristic information is constructed:

$$\eta_{ij}(t) = s_{ij}l_{ij}(1 - b_{ij}), \quad (3)$$

where s_{ij} is the number of points which the new sensor will cover, and

$$l_{ij} = \begin{cases} 1 & \text{if communication exists} \\ 0 & \text{if there is not communication} \end{cases} \quad (4)$$

b is the solution matrix and the matrix element $b_{ij} = 1$ when there is sensor on this position otherwise $b_{ij} = 0$. With s_{ij} we try to locally increase the covered points, with l_{ij} we guarantee that all sensors will be connected; with rule $(1 - b_{ij})$ we guarantee that the position is not chosen yet. When $p_{ij} = 0$ for all values of i and j the search stops. Thus, the construction of the solution stops if no more free positions, or all points are covered or new communication is impossible.

4 Related Work

The positioning of nodes in a sensor network has received a notable attention in research. We present in this section a short review of the published research on this topic.

Zhang [15] study the positioning of sensors in a terrain from the point of view of data transmission. They divide the terrain into cells, then analyze how N sensors should be distributed among the cells, in a way that avoids network bottlenecks and data loss.

In their work [2], Biagioni and Sasaki study different regular positioning methods for sensors: square, triangular and hexagonal grids. In each case they deduce the minimum number of sensors required to provide full coverage, and the resulting fault-tolerance, seen as the minimum number of nodes that have to be shut down in order to degrade the network coverage. They observe a tradeoff between node density and fault-tolerance, being the system with highest node density (thus highest number of nodes) the one

with the highest fault-tolerance. In a similar approach, Kar and Banerjee [8] propose systematic placing methods to ensure connected coverage to 2-dimensional regions and sets of points, that approach the minimum number of sensor nodes required and have polynomial execution times.

In [6], Dhillon and Chakrabarty propose two greedy algorithms that select the locations for a sensor network with minimal number of nodes. They use a grid model for the terrain and consider a probabilistic coverage model for the sensors where the probability of coverage for any point by a given sensor decreases exponentially with its distance from the sensor. Their model allows them to include the effect of obstacles and terrain height as well as incorporate an importance factor that gives preference to the coverage of some part of the terrain. However, their model lacks an explicit method to handle network connectivity or energy optimization

Other works study the performances of random node distributions in a terrain.

Heuristic methods have already been used to solve WSN problems involving network lifetime and coverage. Jourdan and de Weck solved an instance of WSN layout using a multi-objective genetic algorithm in [7]. In their formulation a fixed number of ten sensors has to be placed in order to maximize the coverage and the lifetime of the network. Dijkstra's algorithm is repeatedly applied to the resulting topology to determine the number of rounds that can be performed provided each node has a predefined starting energy. Though the results obtained are encouraging, the small size of the network and the fact that the number of nodes is fixed instead of an optimizable value leave room for further research, as they state in their work.

We will contribute with this work to improve the state-of-the-art of the use of metaheuristics for solving the WSN layout problem. Our aim is to provide an efficient solving method by comparing a set of state-of-the-art metaheuristic techniques applied in the same scenario. We want to solve a new flexible instance in which, for the first time (to the best of our knowledge), both the number and positions of the sensors can be freely chosen, with full coverage of the sensor field guaranteed, and treating the energy efficiency and the overall cost of the network. Besides this, our interest is to tackle complex instances in which the WSN size is in the same order of magnitude as real WSN, with several hundred nodes.

5 Experimental Results

With our algorithm we can solve WSN layout problem on any rectangular area. In this work we solve an WSN problem instance where a terrain of 500×500 meters has to be covered using nodes with coverage and communication radii equal to 30 meters. The terrain has an area of 250,000 square meters, and each sensor covers 2,827 square meters, meaning that in ideal conditions only 89 would be necessary. Now, these ideal conditions do not exist since they would imply that no overlap exists between any two nodes sensing areas, which is impossible due to their geometrical shape (circle). Therefore, the expected minimum number of nodes for full-coverage is higher than 89. An example of solution that achieves full coverage of the region is a square grid formed by the sensors separated by 30 meters. Starting at the HECN, 250 meters have to be

covered to each side of the terrain, requiring 8 sensors. Therefore the grid has 17 ($8 + 8 + 1$) rows and 17 columns, thus 289 sensors including the HECN. In this symmetrical configuration there are four nodes directly connected to the HECN, so the complete traffic of the network 288 messages per round is evenly divided among them. This result is used for comparison. We apply MAX-MIN ant algorithm with the following parameters: $\alpha = \beta = 1$, $\rho = 0.5$, the number of used ants is 3 and the maximum number of iterations is 10. In Table 1 are reported best found results (minimal number of sensors) achieved by several metaheuristic methods. We compare our ACO algorithm results with results obtained by the evolutionary algorithms in [9] and the symmetric solution.

Table 1. Experimental results

Algorithm	Number of sensors
Symmetric	289
MOEA	260
NSGA-II	262
IBEA _{HD}	265
ACO	232

We observe that the ACO algorithm outperforms the symmetric solution and the evolutionary algorithms. We perform 30 independent runs of the ACO algorithm and the achieved numbers of sensors are in the interval $[232, 247]$. The ACO algorithm outperforms the evolutionary algorithms, because the worst found number of sensors by ACO is less than the best found by the evolutionary algorithms.

The ACO is a constructive method, which is managed by pheromone updating and heuristic information. The heuristic information includes the apriory knowledge of the

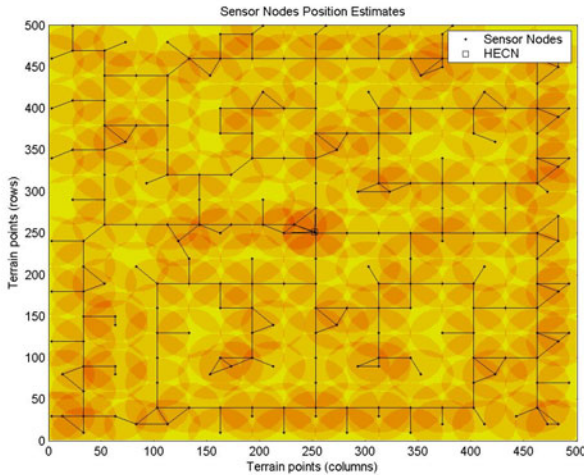


Fig. 2. ACO solution

problem and is one of the crucial point of any ACO algorithms. Our heuristic information is constructed thus to represent the specificity of the problem. So the positions which possibly minimize the number of sensors and in the same time keep the connectivity, becomes more desirable. On another side the genetic algorithm starts from population of random solutions and recombine and mutate them with aim to be improved without taking in to account the specificity of the problem. Thus we can explain the better performance of ACO algorithm.

The ACO solution is represented on Figure 2. With black dots are represented the sensors and with the rings are represented the coverage and connectivity area by a sensor. We can observe there the coverage of the region, positioning of the sensors and connectivity of the network.

6 Conclusions

We have defined a coverage problem for wireless sensor networks with its connectivity constraint. A very large instance consisting of 500×500 square meter area has to be covered using sensors nodes whose sensing and communication radii are 30 meters. We propose ACO algorithm to solve this problem and we compare it with existing evolutionary algorithms. The ACO algorithm outperforms the evolutionary algorithms. The worst found solution by ACO is better than the best found solution by evolutionary algorithms. In a future work we plane to redefine the problem so as to be able to solve more complex WSN layout problem with regions in a sensing area where to put sensors is forbidden and network problem with obstacles. Other interesting direction is to study the robustness of the solutions, to minimize the disturbance in the network when single sensor fail and thus to avoid segmentation of the network.

Acknowledgements. This work has been partially supported by the Bulgarian National Scientific Fund under the grants "Modeling Processes with fixed development rules" DID 02/29 and "Effective Monte Carlo Methods for large-scale scientific problems" DTK 02/44, and by Spanish Ministry of Science and Innovation and FEDER under contract TIN2008-06491-C04-01 (M-project, <http://mstar.lcc.uma.es>). It has also been partially funded by the Andalusian Government under contract P07-TIC-03044 (DIRICOM project, <http://diricom.lcc.uma.es>).

References

1. Alba, E., Molina, G.: Optimal Wireless Sensor Network Layout with Metaheuristics: Solving a Large Scale Instance. In: Lirkov, I., Margenov, S., Waśniewski, J. (eds.) LSSC 2007. LNCS, vol. 4818, pp. 527–535. Springer, Heidelberg (2008)
2. Biagioni, E., Sasaki, G.: Wireless Sensor Placement for Reliable and Efficient Data Collection. In: Proc. Hawaii Int. Conf. Sys. Sci. (2003)
3. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York (1999)
4. Cahon, S., Melab, N., Talbi, E.-G.: Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *J. of Heuristics* 10(3), 357–380 (2004)

5. Deb, K., Pratap, A., Agrawal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: Nsga-ii (2000)
6. Dhillon, S., Chakrabarty, K.: Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks. In: Proc. IEEE Wirel. Comm. Netw. Conf., pp. 1609–1614 (2003)
7. Jourdan, D.B.: Wireless Sensor Network Planing with Application to UWB Localization in GPS-denied Environments. PhD Thesis. Masachusets Institut of Technology (2000)
8. Kar, K., Banerjee, S.: Node Placement for Connected Coverage in Sensor Networks. In: Proc. WiOpt (2003)
9. Molina, G., Alba, E., Talbi, E.-G.: Optimal Sensor Network Layout Using Multi-Objective Metaheuristics. *J. Universal Computer Science* 14(15), 2549–2565 (2008)
10. Nemeroff, J., Garcia, L., Hampel, D., DiPierro, S.: Application of sensor network communications. In: MILCOM 2001, Communications for Network-Centric Operations: Creating the Information Force, pp. 336–341. IEEE (2001)
11. Paek, J., Kothari, N., Chintalapudi, K., Rangwala, S., Govindan, R.: The Performance of a Wireless Sensor Network for Structural Health Monitoring (2004)
12. Stutzle, T., Hoos, H.H.: MAX-MIN Ant System. *J. Future Generation Computer Systems* 16, 889–914 (2000)
13. Werner-Allen, G., Lorinez, K., Welsh, M., Marcillo, O., Jonson, J., Ruiz, M., Lees, J.: Deploying a wireless sensor network on an active volcano. *IEEE J. of Internet Computing* 10(2), 18–25 (2006)
14. Yuce, M.R., Ng, S.W., Myo, N.L., Khan, J.Y., Liu, W.: Wireless body sensor network using medical implant band. *J. Medical Systems* 31(6), 467–474 (2007)
15. Zhang, X., Wicker, S.B.: On the Optimal Distribution of Sensors in a Random Field. *ACM Trans. Sen. Netw.* 1(2), 301–306 (2005)
16. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiño, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)

Countering Evolutionary Forgetting in No-Limit Texas Hold'em Poker Agents

Garrett Nicolai¹ and Robert Hilderman²

¹ Dalhousie University, Halifax, NS, B3H 3J5, Canada

² University of Regina, Regina, SK, S4S 0A2, Canada

Garrett.Nicolai@dal.ca, Robert.Hilderman@uregina.ca

Abstract. No-Limit Texas Hold'em Poker is a stochastic game of imperfect information. Each player receives cards dealt randomly and does not know which cards his opponents have been dealt. These simple features result in No-Limit Texas Hold'em Poker having a large decision space in comparison to other classic games such as Backgammon and Chess. Evolutionary algorithms and neural networks have been shown to find solutions in large and non-linear decision spaces and have proven to aid decision making in No-Limit Texas Hold'em Poker. In this paper, a hybrid method known as evolving neural networks is used by No-Limit Texas Hold'em Poker playing agents to make betting decisions. When selecting a new generation of agents, evolutionary forgetting can result in selecting an agent with betting behaviour that has previously been shown to be inferior. To prevent this from occurring, we utilize two heuristics: halls of fame and co-evolution. In addition, we evaluate agent fitness using three fitness functions based upon, respectively, the length of time an agent survives in a tournament, the number of hands won in a tournament, and the average amount of money won across all hands in a tournament. Results show that the length of time an agent survives is indeed an appropriate measure of fitness. Results also show that utilizing halls of fame and co-evolution serve to further improve the fitness of agents. Finally, through monitoring the evolutionary progress of agents, we find that the skill level of agents improves when using our evolutionary heuristics.

1 Introduction

In the field of Artificial Intelligence, games have attracted a significant amount of research. Games are of interest to researchers due to their well defined rules and success conditions. Furthermore, game-playing agents can be easily benchmarked, as they can play their respective games against previously-created agents, and an objective skill level can be determined.

Successful agents, capable of beating the best human players have been developed for deterministic parlour games such as Chess [8,9] and Checkers [16,17], and stochastic games such as Backgammon [19].

These games all have one key aspect in common: they all involve perfect information. That is, all players can see all information relevant to the game state at all times. Recently, games of imperfect information, such as Poker [12,4,11], and particularly Limit Texas Hold'em [14,11], have started to attract attention in the research community. Unlike Chess and Checkers, where all information is available to all players, Poker involves

deception and hidden information. Part of the allure of card games is that a player must take risks based on incomplete information.

In this paper, we present an algorithm for creating an agent to play a variant of Poker known as No-Limit Texas Hold'em [2,7], which allows any size of bet. Rather than reduce the decision space, we use evolutionary algorithms [1,2,10,12,13,14,16,17,20,19] to teach our agents a guided path to a good solution. Using evolutionary neural networks and iterative play, our agents learn to play No-Limit Texas Hold'em.

2 Rules of No-Limit Texas Hold'em

No-Limit Texas Hold'em is a community variant of the game of Poker. Each player is dealt two cards, referred to as *hole cards*. After the hole cards are dealt, a round of betting commences, whereby each player can make one of three decisions: *fold*, where the player chooses to stop playing for the current round; *call*, where the player chooses to match the current bet, and keep playing; and *raise*, where the player chooses to increase the current bet. This is where No-Limit Texas Hold'em differs from the Limit variant. In Limit Texas Hold'em, each round has a maximum bet. In No-Limit Texas Hold'em, any player may bet any amount, up to and including all of his remaining money, at any time. After betting, three community cards, collectively known as *the flop*, are dealt. The community cards can be combined with any player's hole cards to make the best 5-card Poker hand. After the flop, another betting round commences, followed by a fourth community card, *the turn*. Another betting round ensues, followed by a final community card, known as *the river*, followed by a final betting round. If, at any time, only one player remains due to the others folding, this player is the winner, and a new round commences. If there are at least two players remaining after the final betting round, a *showdown* occurs: the players compare their hands, and the player with the best 5-card Poker hand is declared the winner.

3 Related Work

Research into computer Poker has progressed slowly in comparison with other games, so Poker does not have as large an established literature.

3.1 Limit Texas Hold'em Poker

The Computer Poker Research Group at the University of Alberta is the largest contributor to Poker research in AI. The group recently created one of the best Poker-playing agents in the world, winning the 2007 Poker Bot World Series [11].

Beginning with Loki [3], and progressing through Poki [4] and PsOpti [5], the University of Alberta has concentrated on creating Limit Texas Hold'em Poker players. Originally based on opponent hand prediction through limited simulation, each generation of Poker agents from the UACPRG has modified the implementation and improved upon the playing style of the predecessors. The current agents [11,18] are mostly game theoretic players that try to minimize loss while playing, and are capable of defeating weak to intermediate human players.

3.2 No-Limit Texas Hold'em Poker

No-Limit Texas Hold'em Poker was studied in [7], where a rule-based system was used to model players. The earliest agents were capable of playing a very simple version of two-player No-Limit Texas Hold'em Poker, and were able to defeat several benchmark agents. After modifying the rules used to make betting decisions, the agents were again evaluated, and were shown to have maintained their level of play, while increasing their ability to recognize and adapt to opponent strategies.

No-Limit Texas Hold'em Poker agents were developed in [2], and were capable of playing large-scale games with up to ten players at a table, and tournaments with hundreds of tables. Evolutionary methods were used to evolve betting strategies that considered hand strength and cost. The system begins with some expert knowledge (what was called a head-start approach). Agents were evolved that play well against benchmark agents, and it was shown that agents created using both the evolutionary method and the expert knowledge are more skilled than agents created with either evolutionary methods or expert knowledge.

3.3 Games and Evolutionary Neural Networks

Applying evolutionary algorithms to games is not without precedent. As early as the 1950's, the concept of self-play (i.e., the process of playing agents against themselves and modifying them repeatedly) was being applied to the game of Checkers [16]. In [19] evolutionary algorithms were applied to the game of Backgammon, eventually evolving agents capable of defeating the best human players in the world. In [13], an algorithm similar to that described in [19] was used in conjunction with self-play to create an agent capable of playing small-board Go. In [10][2][20], evolutionary techniques are applied to Chess, improving strategies through iterative play.

Evolutionary methods have also been applied to Poker. In [1], agents are evolved that can play a shortened version of Limit Texas Hold'em Poker, having only one betting round. Betting formulas are evolved by adding and removing parameters, and changing variable weights. Evolution is found to improve the skill level of the agents, allowing them to play better than agents developed through other means.

4 Methodology

Our agents use a 35-20-3 feedforward neural network to learn how to play No-Limit Texas Hold'em. This type of network has three levels, the input level, the hidden level, and the output level. Thirty-five values, which will be explained in Sect. 4.1, are taken from the current game state. These values are combined and manipulated using weighted connections to twenty nodes on the hidden level of the network. The values in the hidden nodes are further manipulated, and result in three values on the output level.

4.1 Input to the Neural Network

The input to the network consists of 35 factors that are deemed necessary to the evaluation of the current state of the poker table, as seen in Table 1.

Table 1. Input to the Neural Network

Input	Feature
1	Chips in pot
2	Chips to call
3	Number of opponents
4	Percentage of hands that will win
5	Number of hands until dealer
6 to 15	Chip counts
16 to 25	Overall Agressiveness
26 to 35	Recent Agressiveness

The Pot. The first feature the value that the agent can win if it wins the hand, and is less than or equal to the total of all of the chips in the pot. If an agent bet all of its chips previously, and betting continued with other agents, it is possible that the current agent is unable to win all of the chips in the pot.

The Bet. The second input feature is the amount of chips that an agent must pay to call the current bet. If another agent has made a bet of \$10, but the current agent already has \$5 in the pot, this value will be \$5. Together with the pot, the bet forms the *pot odd*, a regularly used feature of Poker equal to the ratio of the pot to the bet.

The Opponents. The third input feature is the number of opponents remaining in the hand. As the number of opponents increases, it becomes harder to win a hand, and thus, the agent must become more selective of the hands that it decides to play.

The Cards. The fourth input to the neural network is the quality of the cards that the agent is holding. The quality of an agent's cards is dependent upon two factors: *hand strength*, and *hand potential*. Hand strength represents the likelihood that a hand will win, assuming that there will be no cards to come. Hand potential represents the likelihood that a hand will improve based upon future cards. For example, after the hole cards are dealt, a pair of fours would have good hand strength, but poor hand potential. At this point, only ten hands can beat it, namely the ten higher pairs; however, when further cards are played, there are many more potential better hands.

Before any evolutionary trials were run, an exhaustive set of lookup tables were built. These lookup tables can quickly report the likelihood that a hand will win, should a showdown occur. Entries are calculated for all possible situations of the game, with any number of table opponents from 1 to 9. Exhaustive simulations were run to calculate the percentage of hands that an agent would win, given its hole cards, and the current situation of the game.

The lookup tables were divided into three states of the game: pre-flop, post-flop, and post river. For the post-turn stage of the game, the post-river tables were used, looped for each possible river card, and calculations were made at run-time. The pre-flop table was a two-dimensional matrix, representing the 169 potential hole card combinations, marked for each possible number of opponents. The pre-flop table has 1,521 total entries.

The post-flop stage requires multiple tables, all of which are 3-dimensional matrices. The first two dimensions contain the number of hole card combinations and opponents, respectively. In the pre-flop stage, suits were unimportant, but flushes are now possible, and suits must be recorded. The third dimension represents the number of potential flops of a particular type. Flops are sub-divided into five categories: `ONE_SUIT`, where all three community cards are of the same suit; `TWO_SUIT`, where the three community cards fall into one of two suits; `THREE_SUIT`, where all of the community cards are of different suits and ranks; `THREE_SUIT_DOUBLE`, where the suits are different, but two cards have the same rank; and `THREE_SUIT_TRIPLE`, where the suits are all different, but the ranks are all the same.

The post-river tables are again 2-dimensional, discarding the differences for different opponent numbers. Since all cards have been played, winning percentage can be calculated quickly at run-time. The post-river tables are divided into five sub-groups: `FIVE_SUITED`, where all five community cards are of the same suit; `FOUR_SUITED`, where four cards are of one suit, and the other is another suit; `THREE_SUITED`, where three cards are of one suit, and the other two are of other suits; and `NO_SUITED`, where less than three cards are of the same suit, and thus flushes are not possible. The `BUILD_1_SUIT` algorithm gives an example of how the flop tables are generated.

```

1: procedure BUILD_1_SUIT
2:   begin
3:     FlopID = 0
4:     for i = TWO to ACE do
5:       Flop[0] = Card(i,0)
6:       for j = i + 1 to ACE do
7:         Flop[1] = Card(j, 0)
8:         for k = j + 1 to ACE do
9:           Flop[2] = Card(k, 0)
10:        HoleID = 0
11:        for m = TWO to ACE * 2 do
12:          if m in Flop continue
13:          for n = m + 1 to ACE * 2 do
14:            if n in Flop continue
15:            if m < ACE then
16:              Hole[HoleID][0] = Card(m, 0)
17:            else Hole[HoleID][0] = Card(m,1)
18:            if n < ACE then
19:              Hole[HoleID][1] = Card(n, 0)
20:            else Hole[HoleID][1] = Card(n, 1)
21:            HoleID++;
22:            for m = TWO to ACE do
23:              for n = TWO to ACE do
24:                Hole[HoleID][0] = Card(m, 1)
25:                Hole[HoleID++][1] = Card(n, 2)
26:              endfor
27:            endfor
28:            BUILD_ROW(Table1Suit[FlopID], Hole, HoleID, Flop)
29:            FlopID++;
30:          end for
31:        end for
32:      end for
33:    end BUILD_1_SUIT

```

The first 10 lines loop through the possible cards for the flop, creating each potential flop of one suit. Lines 11 through 20 cover 3 cases of hole cards: the hole cards are of the same suit, and it is the same suit as the flop; the hole cards are of the same suit, and it is not the same suit as the flop; and the hole cards are different suits, but one of the cards is the same suit as the flop. Lines 22 to 27 cover the remaining case: the hole cards are of different suits, and neither card is the same suit as the flop. The `BUILD_ROW` function shown on line 28 is used to loop through all potential opposing hands, and return a percentage of hands that will win if the hand is played all the way to a showdown. The other functions to build tables work similarly.

The Position. In Poker, it is desirable to maximize information about opponents by having them bet before you do. This input value starts at 0, when the agent is the last bettor in a round. After the round, the value resets to the number of players at the table, and decreases by one for each round that is played. Thus, the value will be equal to the number of rounds remaining until the agent is betting last.

The Chips. The next inputs to the network are public information, and are known by all of the players. This input is relative to the current agent, and will shift depending upon its seat. Input 6 will always be the number of chips of the agent making the decision, input 7 will be the chip count of the agent in the next seat, and so on.

It is important to know the remaining chips of each particular agent that is playing in a particular round, as it will affect their decisions. An opponent with less chips is less likely to call a big raise, and it might be desirable to play aggressively to steal its chips. It is also important to keep track of the chip counts in relation to an agent's position. If an agent is sitting next to another agent with many chips, it may make sense to play a little more conservative, as the larger chip stack can steal bets with large over-raises.

Aggressiveness. The final twenty inputs to the neural network are concerned with opponent modeling. Since there is so much hidden information, the agent must use whatever it can to try to determine the quality of its opponents' hands. The only information that an invisible opponent gives away is its betting strategy.

However, it is not as simple as determining that a raise means that an opponent has good cards. Opponents try to disguise their cards by occasionally betting counter to what logic might dictate. Our agents are capable of bluffing, as discussed in Sect 4.3. Luckily, there are a few pieces of knowledge that an agent can use to its advantage to counteract bluffing.

All other things being equal, cards are stochastic. In the long run, certain hands will occur with known probability, and an opponent's actions can be compared to that probability. If an opponent is betting more often than probability would dictate, it can be determined that the opponent is likely bluffing, and its high bets can be adjusted to compensate. Likewise, an agent will be more wary when an opponent that never bets begins calling and raising.

All opponents' bets are recorded over the long term and the short term according to equation 1. The bets are simplified to a single value. If an opponent folds, that opponent receives a value of 0 for that decision. If an opponent calls, that opponent receives a value of 1 for that decision, and if an opponent raises, then that opponent receives a value equal to the new bet divided by the old bet; this value will always be greater than 1.

$$\text{Aggressiveness} = \frac{\text{BetAmount}}{\text{CallAmount}} \quad (1)$$

Aggressiveness over the Long-Term. The aggressiveness values are a running average of the decisions made by any particular agent. For example, an agent might have an aggressiveness of 1.3 over 55 decisions made. This could be seen as an agent that generally calls, but occasionally raises. The agent has a cumulative aggressiveness value of 1.3×55 , or 71.5. If in the next hand, the agent calls the bet, and then folds, it will get values of 1.0 and 0.0 for its call and fold, respectively. Its aggressiveness will now be 72.5 over 57 decisions, giving a score of 1.27.

The aggressiveness vectors are an attempt to model opponent tendencies, and take advantage of situations where they play counter to these tendencies. For example, if an opponent with low aggressiveness suddenly makes a large bet, it can be inferred that either the opponent has really good cards, or is making a very large bluff, and the deciding agent can react appropriately. The agents also keep track of their own aggressiveness, with the goal of preventing predictability. If an agent becomes too predictable, they can be taken advantage of. Agents can then make decisions counter to their decisions to throw off opponents.

Aggressiveness over the Short-Term. Although agents will generally fall into an overall pattern, it is possible to ignore that pattern for short periods of time. Thus, agents keep track of short-term aggressiveness of their opponents. Short-term aggressiveness is calculated in the same way as long-term aggressiveness, but only over the last ten hands of a particular tournament. Ten hands is enough for each player to have the advantage or disadvantage of betting from every single position at the table.

For example, an opponent may have an overall aggressiveness of 1.6, but has decided to play more conservatively over the last 10 hands, having an aggressiveness of 0.5 over these hands. Although this agent can be expected to call or raise a bet, recently, they are as likely to fold to a bet as they are to call. Whereas the long-term aggressiveness values might indicate that a raise would be the best decision, the short-term aggressiveness might suggest a call instead.

4.2 The Hidden Layer

The hidden layer of the neural network consists of twenty nodes that are fully connected to both the input and output layers. Twenty nodes was chosen early in implementation, and may be an area for future optimization.

4.3 The Output Vector

In Sect. 4 it was stated that the output layer consisted of three nodes. These nodes correspond to a fold, a call or a raise. Raises are further divided into small, medium, and large raises, which will be explained later in this section. The output of the network is stochastic, rather than deterministic; rather than choosing the decision with the highest likelihood, any decision can be made with a certain percentage. By occasionally making sub-optimal choices, the agent can disguise its playing style.

Raises are distinguished into small raises, medium raises, and large raises. After observing many games of Texas Hold'em, it was determined that the biggest determiner of whether a raise was considered small, medium, or large was the percentage of a player's chip count that a bet made up. Bets that were smaller than 10% of a player's chips were considered small, bets that were larger than a third of a player's chips were considered large, and bets that were in between were considered medium.

Again, bluffing was encouraged, and bets were not restricted to a particular range. Although a small bet might be 10% of an agent's chips, we allowed the potential to make larger (or smaller) bets than the output vector might otherwise allow. An agent might bluff all of its chips on a recommended small bet, or make the smallest possible bet when a large bet was suggested. Watching television and internet Poker, it was determined that generally, players are more likely to make a bet other than the recommended one when they are sure of their cards; when the cards are good, players become more creative in betting.

The bets are determined using a normal distribution, centred around the values shown in Table 2.

Table 2. Values used in GetBet algorithm

Value	Description
0.06	LoUpper
0.7	LoInside
0.1	MedLower
0.2	MedUpper
0.6	MedInside
0.1	MedOutLo
0.3	MedOutHi
0.3	HiPoint
0.95	HiAbove
0.05	HiBelow
0.1	HiAllIn

Thus, small bets normally fall in the range of 0 to LoUpper, that is, 6% of an agent's chips. However, this only occurs with a likelihood of 70%. The other 30% of the time, a small bet will be more than 6% of an agent's chips, with a normal distribution with a mean at 6%. The standard deviation is calculated such that the curves for the standard and non-standard bets are continuous.

Medium bets are standard within a range of 10 and 20% of an agent's chips, 60% of the time. 10% of medium bets are less than 10% of an agent's chips, while 30% are more than 20% of an agent's chips, again with a normal distribution.

High bets are centred around 30% of an agent's chips. 5% of the time, the bet will be less, while 95% of large bets will be more than 30% of an agent's chips, with the top 5% of high bets consisting of all of an agent's chips. If an agent is betting a high percentage of its chips, it should bet all of them. If it loses the hand, it is as good as eliminated anyway, and thus bets all instead of almost all of its chips. It is better to risk the chips on a good hand than to be forced to lose the rest of the chips on a forced bet.

4.4 Evolution

Evolutionary algorithms model biological evolution. Agents compete against each other, and the fittest individuals are chosen for reproduction and further competition. The EVOLUTION Algorithm demonstrates the selection of fittest individuals in a population.

```

1: procedure EVOLUTION(Generations, NumPlayers, NumPlayersKept,
                       Tournaments)
2: begin
3:   for i = 0 to NumPlayers - 1 do
4:     Players[i] = new Player(Random)
5:   end for
6:   for i = 0 to Generations - 1 do
7:     for j = 0 to Tournaments - 1 do
8:       PlayTournament()
9:     endfor
10:    SortPlayers(Players)
11:    for j = 0 to NumPlayersKept - 1 do
12:      KeptPlayers[j] = Players[j];
13:    end for
14:    EVOLVE_PLAYERS(Players, KeptPlayers, NumPlayers,
                    NumPlayersKept)
15: end for
16: end EVOLUTION

```

In lines 3 and 4, *NumPlayers* agents are created with neural networks with random weights between -1 and 1. These agents begin playing tournaments. Decisions are made by the individuals using the input and output of the neural networks described in Sects. [4.1](#) and [4.3](#). Tournament are sub-divided into tables, each of which hosts ten agents. After each round of Poker, the tournament is re-organised to minimize the number of tables. Any agents that have been eliminated have their finishing positions recorded, and the process begins again. After *Tournaments* number of tournaments have been completed, the agents are sorted according to their average ranking. The *numPlayersKept* best agents are then supplied to the EVOLVE_PLAYERS algorithm, which will create new agents from the best agents in this generation. In order to preserve the current results, the best agents are kept as members of the population for the next generation, as shown in lines 11 and 12.

```

1: procedure EVOLVE_PLAYERS(Players[], Elite[], NumPlayers,
                           NumPlayersKept[])
2: begin
3:   ParentCount = 1
4:   for i = NumPlayersKept to NumPlayers do
5:     if numPlayersKept == 1 then
6:       Players[i] = new Player[Elite[0]]
7:     else
8:       ParentCount = Random.Exponential()
9:       Parents = new NeuralNet[ParentCount]
10:      //Choose parents from Elite
11:      for j = 0 to ParentCount - 1 do
12:        Weights[j] = Random.UniformDouble()
13:      end for
14:      normalise(Weights)
15:      for j = 0 to NumLinks do
16:        Value = 0
17:        for k = 0 to ParentCount do
18:          Value += Parents[k].links[j] x weights[k]
19:        endfor
20:        Players[i].links[j] = Value
21:        random = Random.UniformDouble()
22:        if random < mutationLikelihood then
23:          Players[i].links[j] += Random.Gaussian(mutMean, mutDev)
24:        end if
25:      end for
26:    end for
27: end EVOLVE_PLAYERS

```

The EVOLVE_PLAYERS algorithm describes the creation of new agents for successive generations in the evolutionary algorithm. In lines 8 through 12, parents are chosen from the best agents of the previous generation. Our agents are not limited to two parents; their parents may be comprised of all of the elite agents from the previous generation. Once the parents are selected, they are given weighted importance, influencing how much a child will resemble each parent. The values of the child's neural network are calculated as a weighted sum of the parent's network links. For example, if an agent has two parents, weighted at 0.6 and 0.4, and the parent links between two nodes are 1 and -1, respectively, then the new agent's link value would be 0.2, calculated as $0.6 * 1 + 0.4 * -1$.

However, if the child agents are simply derived from the parents, the system will quickly converge. A mutation factor is introduced in line 21 to promote exploration. After child's network values have been assigned, random noise is applied to the weights, with a small likelihood.

4.5 Alternate Fitness Functions

At each generation, the agents that were chosen for replication were those that lasted the longest in the tournaments. Intuitively, it was determined that these agents would be the best in their respective generations. It is possible, however, that these agents survived not due to good Poker skills, but rather because they were overly conservative.

Two alternate fitness functions were also used to varying degrees to determine if tournament survival was the optimal fitness function.

The MeanMoney fitness function would rank agents according to the average amount of money that they won across all of the hands that they played. The HandsWon fitness function ranked agents according to how many hands they had won. The HandsWon function was combined with the original tournament success function in a series of increments. In the pure HandsWon function, the ranking of the agents for selection is solely dependent upon the number of hands that they have won. At Hands80, the ranking is weighted 80% towards the number of hands won, and 20% towards their ranking in the tournaments. Likewise, Hands60, Hands40 and Hands20 were also investigated. All of the agents evolved with alternate fitness functions were otherwise identically evolved to the original agents, with no evolutionary counter-measures.

4.6 Evolutionary Forgetting

In [15], it is suggested that evolutionary algorithms can occasionally get caught in less-than-optimal loops. In this case, agent A is deemed to be the best of a generation, is replaced by B in the next generation, which in turn is defeated by an A-type agent in the subsequent generation. In [14], it is suggested that an evolutionary system can lose its learning gradient, or fall prey to *Evolutionary Forgetting*. Evolutionary forgetting occurs when a strategy is promoted, even when it is not better than strategies of previous generations.

For example, in Poker, there is a special decision strategy known as a *check-raise*. It involves making a call of \$0 to tempt opponents to make a bet. Once the opponent makes a reasonable bet, the player raises the bet, often to a level that is not affordable to the opponents. The opponents fold, but the player receives the money that they bet. A check-raise strategy may be evolved in an evolutionary Poker system, and for several generations, it may be the strongest strategy. However, once a suitable counter strategy is evolved, the check-raise falls into disuse. Since the check-raise is no longer used, strategies no longer need to defend against it, and the strategies, although seeming to improve, forget how to play against a check-raise. Eventually, the check-raise may surface again, and because current strategies do not defend against it, it is seen as superior. This cycle can continue indefinitely, unless some measure is implemented to counter evolutionary forgetting. Several strategies exist for countering evolutionary forgetting, as presented in [15], but are suggested for two-player games. We have adapted these strategies for evolutionary Poker, which can contain up to ten players per table, and thousands of players in tournaments.

4.7 Halls of Fame

A hall of fame serves as a genetic memory for an evolutionary system, and can be used as a benchmark of previous generations. Agents in the hall of fame are not used to create new agents. Their sole purpose in the population is as a competition benchmark for the competing agents. As long as the regular agents are competing against the hall of fame agents, their strategies should remember how to defeat the old strategies, and thus promote steady improvement.

The hall of fame begins with no agents included. After the first tournaments are played, the agents that are selected for reproduction are also inserted into the hall of fame. In the next generation, the playing population will consist of the regular population of agents, as well as the hall of fame agents. Here, a decision must be made. It is possible to create a very large hall of fame, as memory permits, but this quickly becomes computationally expensive. The population size of the evolutionary system will increase regularly. Given that many hands are required in each tournament to eliminate all of the agents, as the population size grows, so too does the time required per tournament, and hence, per generation.

Our hall of fame was had a fixed size, and could include no more agents than were in the original population. The best agents of previous generations would still be present in the hall of fame, but the size of the hall would not quickly get out of hand. After each generation, agents in the hall of fame were replaced according to the REPLACE algorithm.

```

1: procedure REPLACE(HallOfFame[], hallSize, Players[],
                    numPlayersKept)
2: begin
3:   j = 0;
4:   for i = 0 to numPlayersKept - 1 do
5:     if HallOfFame[hallSize - numPlayersKept + i].OverallRanking()
       > Players[j].OverallRanking() then
6:       HallOfFame[hallSize - numPlayersKept + i] = Players[j++]
7:     else continue
8:     end if
9:   end for
10: end REPLACE

```

The Players and HallOfFame must be sorted before calling REPLACE. numPlayersKept is the amount of agents that are selected for reproduction in a given generation. As long as the rank of the xth best agent is lower (i.e. better) than the x-lowest hall of fame member the member is replaced. In the worst case, when all agents in the hall of fame are replaced every generation, the hall of fame will still have a memory of 10 generations. The memory is generally much longer.

4.8 Co-evolution

In [13][4][15], it is suggested that co-evolutionary methods may counter evolutionary forgetting. In co-evolution, several independent populations are evolved simultaneously. Each population has its own set of agents, and when reproduction occurs, the eligible agents are chosen from the individual populations. By evolving the populations separately, it is hoped that each population will develop its own strategies.

Multiple populations are created in the same way as if there were only a single population. They are then allowed to compete together, similarly to how the agents can compete against agents in a hall of fame. When it comes time for selection, agents are only ranked against agents in their respective populations. It is possible that one population may have a superior strategy, and that the agents from this population out-rank all agents from all other populations. Regardless, agents are separated by population for

evaluation and evolution, in order to preserve any unique exploration paths that alternate populations might be exploring.

Like halls of fame, the main strategy of co-evolution is a deepening of the competition. By having separately evolving populations, the agents are exposed to a more varied set of strategies, and thus can produce more robust strategies. Often, as co-evolution proceeds, a situation known as an *arms race* will develop. In an arms-race, one population develops a good strategy, which is later supplanted by another population's counter-strategy, which then repeats. As each population progresses, the global skill level also increases. Agents are less concerned with defeating strategies that it has already seen, and more concerned with defeating new strategies as they come along.

Halls of fame can be added to co-evolution. Our system gives each population its own hall of fame, with the same replacement strategy as when there is only one population. As stated in Sect. 4.7 the goal of the halls of fame was to protect older strategies that might get replaced in the population. In a co-evolutionary environment, it is entirely possible that one sub-population may become dominant for a period of several generations. If a global hall of fame is used, the strategies of the weaker populations would quickly be replaced in the hall of fame by the strategies of the superior population. Each population was given its own hall of fame to preserve strategies that might not be the strongest in the larger population, but could still be useful competitive benchmarks for the evolving agents.

4.9 Duplicate Tables

Poker is a game with a high degree of variance. Skill plays a large part in the determination of which players will win regularly, but if a good player receives poor cards, he will most likely not win. In [6], a method for evaluating agents is discussed, which is modeled upon the real-world example of *duplicate tables*. In professional Bridge tournaments, duplicate tables are used to attempt to remove some of the randomness of the cards. Unfortunately, due to the relatively high cost of performing duplicate tables at each hand, they are not used in the evolution process. We only use duplicate tables after the evolution has been completed, as a method to test our best evolved agents against certain benchmarks.

A duplicate table tournament is a collection of the single tournaments described in Sect. 4.4. An agent sits at a table, and the tournament is played until every agent at the table has been eliminated (i.e., every agent except one has lost all of its chips). The rankings of the agents are noted, and the next tournament can begin.

Unlike a normal tournament, where the deck would be re-shuffled, and agents would again play to elimination, the deck is reset to its original state, and each agent is shifted one seat down the table. Thus, the agent at seat 5 is now at seat 6, and so on, with the agent previously at seat 9 now at seat 0. Again, the agents play to elimination. Since the deck was reset, the cards will be exactly the same as they were last time; the only difference will be which agents receive which cards. Each agent keeps no memory of previous tournaments. This method continues until each agent has sat at each seat at the table, and thus had a chance with each possible set of hole cards. After the completion of a revolution of the table and the noting of the rankings, the agents would then play with new cards.

5 Experimental Results

In order to evaluate agents, a number of benchmarks were used. In [2,5,7,18], several static agents, which always play the same, regardless of the situation, are used as benchmarks. These agents are admittedly weak players, but are supplemented by the best agents developed in [2], and can be used to evaluate the quality of our evolved agents relative to each other. The benchmark agents are as follow: Folder, Caller, and Raiser, that always fold, call and raise, respectively, at every decision; Random, that always makes random decisions; CallOrRaise, that calls and raises with equal likelihood; OldBest, OldScratch, OldStart, that were developed in [2]. OldScratch was evolved with no head start to the evolution, OldBest was evolved with a head start, and OldStart was given a head start, but no evolution.

Baseline agents were evolved from a population of 1000 agents, for 500 generations, playing 500 tournaments per generation. After each generation, agents were ranked according to their average. After each generation, the 100 best agents were selected for reproduction, and the rest of the population was filled with their offspring. *LargeHOF* agents were also evolved from a population of 1000 agents, but included a hall of fame of size 1000. *SmallHOF* agents were evolved from a smaller population of 500 agents, with a hall of fame of size 500, and only 50 agents were selected for reproduction each generation. *HOF2Pop* agents were evolved using two co-evolutionary populations of 500 agents each, each with a hall of fame of 500 agents.

After 500 generations, the best agent from the 500th generation played 100,000 duplicate table tournaments, with each of the benchmark agents also sitting at the tables. After each duplicate table tournament, the ranking of each agent at the table was gathered, and the average was calculated after the completion of all 100,000 duplicate table tournaments. There were nine agents at the duplicate tables. The best possible rank was 1, corresponding to an agent that wins every tournament, regardless of cards or opponents. The worst possible rank was 9, corresponding to an agent that was eliminated from every tournament in last place. The results of our best agents are shown in Fig. 1.

Figure 1 represents the results of the duplicate table tournaments. In Fig. 1, the Control agent represents the agent that is being evaluated, while the other vertical bars represent the rankings of the other agents in the evaluation of the control agent. For example, the first bar of *Random* represents how the *Random* agent performed against the *Baseline* agent, the second bar represents how the *Random* agent performed against the *SmallHall* agent, and so on.

The best results were obtained by the agents evolved with a large hall of fame, but no co-evolution. These agents obtained an average rank of 2.85 out of 9. Co-evolution seemed to have little effect upon the agents when a hall of fame was used, and the agents in the two co-evolutionary populations received average ranks of 2.92 and 2.93. The difference between the best agents and the second best seems to be quite small. A two-tailed paired t-test was conducted on the null hypothesis that the ranks of any two distinct agents were equal. In all cases, and for all experiments, the null hypothesis was rejected with 99% confidence. Although the difference is small, enough hands were played that even small differences equate to a difference in skill. The small hall of fame also seemed to have an impact; although the agent was evolved in a smaller population

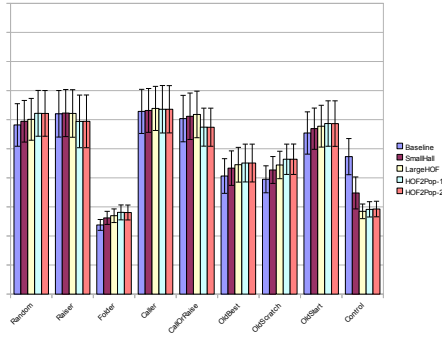


Fig. 1. Results of Duplicate Table Tournaments (Original in Colour)

than the baseline, and thus had less competition, it was able to achieve a rank of 3.48, which was more than one full rank better than the baseline agent’s 4.73.

The baseline agent itself out-performed all of the benchmarks, with the exception of the best agents evolved in [2], and the Folder. The best agents evolved in our experiments out-ranked all of the benchmarks, except for the folder, although the best agents were much closer to the Folder’s rank than the other agents. It was surprising that the Folder performed so well, considering that it makes no decisions, and simply lays down its cards at every decision point. However, in an environment where there are many aggressive players, such as automatic raisers and callers, many of these players will eliminate each other early, giving better ranks to conservative players. The better ranks of our best agents tell us that they can survive the over-active early hands until the aggressive players are eliminated, and then succeed against agents that actually make decisions.

To ensure that the original fitness function was appropriate, we evaluated agents evolved with alternate fitness functions, as described in Sect. 4.5. We see the results of such agents in Fig. 2. Compared with the agents evolved without halls of fame or co-evolution, the alternate fitness functions do not perform as well. The rank obtained by the baseline agent in its duplicate table tournaments was 4.73, which was almost a full rank higher than the 5.48 obtained by the MeanMoney agent, and the 5.49 achieved when the HandsWon algorithm was weighted at 80%. These two rankings were the

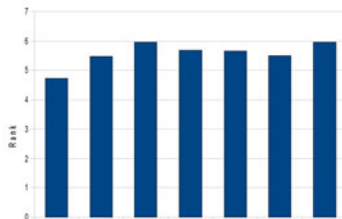


Fig. 2. Results of alternate fitness functions

best obtained by the alternate fitness functions, and were still nowhere near the rank achieved by the baseline agent. It would appear that the tournament selection fitness function is the most appropriate function, as was hypothesized.

6 Evolutionary Progress

In the previous section, the best agents from the 500th generation of evolution were chosen to play in the duplicate table tournaments, based upon the assumption that agents were improving from generation to generation. To test this assumption, we developed *evolutionary progress tournaments*. In an evolutionary progress tournament, the best agent from each generation of an evolutionary trial was inserted into a playing population, which would then play 10,000 tournaments, and their average ranking was calculated. If every agent was exactly equal, each would receive an average ranking of 250, namely half of the number of agents competing. We defined evolutionary progress as either maintenance or improvement of skill level from one generation to the next. As the ranking becomes lower, the agents are improving. The best evolutionary systems would see a monotonic decrease from generation 1 to generation 500.

First, we examine the evolutionary progress of the baseline agents, shown in Fig. 3. These results are for the agents evolved with no evolutionary counter-measures. In Fig. 3 we see that although the final generation does have a better rank than the first generation, the graph is not monotonic. At approximately generation 50, the system hits its best point, and after staying there for a few generations, decreases in skill until about generation 200, where the skill level of the agents finally converges at about 250. Section 4.6 introduced the idea of evolutionary forgetting, and suggested that counter-measures such as a hall of fame and co-evolution might fix such a problem.

Figure 4 shows the evolutionary progress graph for the agents evolved with a hall of fame, but no co-evolution. In Sect. 5 it was determined that these were the best overall agents of all the ones that were evolved. In 4 we see a different picture than that obtained in the graph of the agents evolved without countermeasures. Note that although the ranks in this graph are higher than those in Fig. 3, the agents are actually more skilled; the ranks in Fig 4 have no correlation to those in 3. There is a slight decrease in skill around generation 100, but after generation 200, the agents are generally decreasing. Furthermore, by generation 400, the skill levels become much less noisy, and we still see an improvement in agents right up to generation 500, suggesting that the

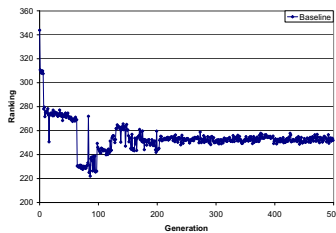


Fig. 3. Evolutionary Progress for Baseline Agents

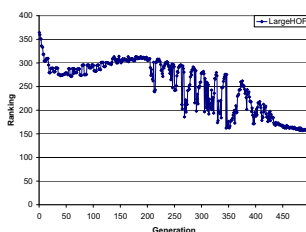


Fig. 4. Evolutionary Progress for Hall of Fame Agents

agents might even improve beyond their current skill level. It would appear that the Hall of Fame is at least partially forcing agents to find better strategies, rather than simply recycle old ones.

7 Conclusions

Our algorithms present a new way of creating agents for No-Limit Texas Hold'em Poker. Previous agents have been concerned with the Limit variant of Texas Hold'em, and have been centered around simulation [45] and game theoretical methods [11,18]. Our approach is to evolve agents that learn to play No-Limit Texas Hold'em through experience, with good agents being rewarded, and poor agents being discarded. Evolutionary neural networks allow good strategies to be discovered, without providing much apriori knowledge of the game state. By making minute changes to the networks, alternative solutions are explored, and agents discover a guided path through an enormous search space. Furthermore, it was determined that while co-evolution and halls of fame had a constructive influence on the results, alternate fitness functions did not.

References

1. Barone, L., While, L.: An adaptive learning model for simplified poker using evolutionary algorithms. In: Proceedings of the Congress on Evolutionary Computation, vol. 1, pp. 153–160 (1999)
2. Beattie, B., Nicolai, G., Gerhard, D., Hilderman, R.: Pattern classification in No-Limit Poker: A head start evolutionary approach. In: Canadian Conference on AI, pp. 204–215 (2007)
3. Billings, D., Papp, D., Pena, L., Schaeffer, J., Szafron, D.: Using selective-sampling simulations in poker. In: AAAI Spring Symposium on Search Techniques for Problem Solving Under Uncertainty and Incomplete Information (1999)
4. Billings, D., Davidson, A., Schaeffer, J., Szafron, D.: The challenge of poker. *Artificial Intelligence* 134, 201–240 (2002)
5. Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T., Szafron, D.: Approximating game-theoretic optimal strategies for full-scale poker. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (2003)
6. Billings, D.: Algorithms and Assessment in Computer Poker. PHD Dissertation. University of Alberta (2006)
7. Booker, L.: A No Limit Texas Hold'em poker playing agent. Master's Thesis. University of London (2004)

8. Campbell, M., Hoane, A., Hsu, F.: Deep Blue. *Artificial Intelligence* 134, 57–83 (2002)
9. Donninger, C., Lorenz, U.: The Hydra project Xcell. *Journal* 53, 94–97 (2005)
10. Hauptman, A., Sipper, M.: GP-EndChess: Using genetic programming to evolve chess endgame players, booktitle. In: *Proceedings of the 8th European Conference on Genetic Programming* (2005)
11. Johanson, M.: Robust strategies and counter-strategies: Building a champion level computer poker player. Master's thesis. University of Alberta (2007)
12. Kendall, G., Whitwell, G.: An evolutionary approach for tuning of a chess evaluation function using population dynamics. In: *Proceedings of the 2001 IEEE Congress on Evolutionary Computation*, pp. 995–1002 (2001)
13. Lubberts, A., Miiikkulainen, R.: Co-evolving a go-playing neural network. In: *Proceedings of the GECCO-01 Workshop on Coevolution: Turning Adaptive Algorithms Upon Themselves* (2001)
14. Pollack, J., Blair, A.: Co-evolution in the successful learning of backgammon strategy. *Machine Learning* 32, 225–240 (1998)
15. Rosin, C.: Coevolutionary search among adversaries. PHD Dissertation. University fo California, San Diego (1997)
16. Samuel, A.L.: Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development* (1959)
17. Schaeffer, J., Cublerson, J., Treloar, N., Knight, B., Lu, P., Szafron, D.: A world championship caliber checkers program. *Artificial Intelligence* 53, 273–289 (1992)
18. Schauenberg, T.: Opponent modelling and search in poker. Master's thesis. University of Alberta (2006)
19. Tesauro, G.: Programming backgammon using self-teaching neural nets. *Artificial Intelligence* 134, 181–199 (2002)
20. Thrun, S.: Learning to play the game of chess. *Advances in Neural Information Processing Systems* 7, 1069–1076 (1995)

Model Regularization in Coevolutionary Architectures Evolving Straight Line Code

César L. Alonso¹, José Luis Montaña², Cruz Enrique Borges²,
Marina de la Cruz Echeandía³, and Alfonso Ortega de la Puente³

¹ Centro de Inteligencia Artificial, Universidad de Oviedo,
Campus de Gijón, 33271 Gijón, Spain
calonso@aic.uniovi.es

² Departamento de Matemáticas, Estadística y Computación
Universidad de Cantabria, 39005 Santander, Spain
cruz.borges@alumnos.unican.es, montanj1@unican.es

³ Departamento de Ingeniería Informática, Escuela Politécnica Superior
Universidad Autónoma de Madrid, Madrid, Spain
{marina.cruz, alfonso.ortega}@uam.es

Abstract. Frequently, when an evolutionary algorithm is applied to a population of symbolic expressions, the shapes of these symbolic expressions are very different at the first generations whereas they become more similar during the evolving process. In fact, when the evolutionary algorithm finishes most of the best symbolic expressions only differ in some of its coefficients. In this paper we present several coevolutionary strategies of a genetic program that evolves symbolic expressions represented by straight line programs and an evolution strategy that searches for good coefficients. The presented methods have been applied to solve instances of symbolic regression problem, corrupted by additive noise. A main contribution of the work is the introduction of a fitness function with a penalty term, besides the well known fitness function based on the empirical error over the sample set. The results show that in the presence of noise, the coevolutionary architecture with penalized fitness function outperforms the strategies where only the empirical error is considered in order to evaluate the symbolic expressions of the population.

Keywords: Genetic Programming, Straight-line Programs, Coevolution, Symbolic Regression, Penalty term.

1 Introduction

Coevolutionary strategies can be considered as an interesting extension of the traditional evolutionary algorithms. Basically, coevolution involves two or more evolutionary processes with interactive performance. Initial ideas on modelling coevolutionary processes were formulated in [1], [2] or [3]. A coevolutionary strategy consists in the evolution of separate populations using their own evolutionary parameters (i.e. genotype of the individuals, recombination operators, ...) but with some kind of interaction between these populations. Two basic classes of coevolutionary algorithms have been

developed: competitive algorithms and cooperative algorithms. In the first class, the fitness of an individual is determined by a series of competitions with other individuals. Competition takes place between the partial evolutionary processes coevolving and the success of one implies the failure of the other (see, for example, [4]). On the other hand, in the second class the fitness of an individual is determined by a series of collaborations with other individuals from other populations.

The standard approach of cooperative coevolution is based on the decomposition of the problem into several partial components. The structure of each component is assigned to a different population. Then the populations are evolved in isolation from one another but in order to compute the fitness of an individual from a population, a set of collaborators are selected from the other populations. Finally a solution of the problem is constructed by means of the combination of partial solutions obtained from the different populations. Some examples of application of cooperative coevolutionary strategies for solving problems can be found in [5] and [6].

This paper focuses on the design and the study of several coevolutionary strategies between Genetic Programming (GP) and Evolutionary Algorithms (EA). Although in the cooperative systems the coevolving populations usually are homogeneous (i.e. with similar genotype representations), in this case we deal with two heterogeneous populations: one composed by elements of a structure named *Straight Line Program* (SLP) that represents programs and the other one composed by vectors of real constants. The coevolution between GP and EA was applied with promising results in [7]. In that case a population of trees and another one of fixed length strings were used.

We have applied the strategies to solve instances of symbolic regression problem. The problem of symbolic regression consists in finding in symbolic form a function that fits a given finite sample set of data points. More formally, we consider an input space $X = \mathbb{R}^n$ and an output space $Y = \mathbb{R}$. We are given a set of m pairs sample $z = (x_i, y_i)_{1 \leq i \leq m}$. The goal is to construct a function $f : X \rightarrow Y$ which predicts the value $y \in Y$ from a given $x \in X$. The empirical error of a function f with respect to z is:

$$\varepsilon_z(f) = \frac{1}{m} \sum_{i=1}^m (f(x_i) - y_i)^2 \quad (1)$$

which is known as the mean square error (MSE).

In this work we have considered noisy regression problem instances with additive gaussian noise. The procedure to construct such kind of problem instances is the following: let $g : X \rightarrow Y$ be the target function used for obtaining the sample points. Then the set $z = (x_i, y_i)_{1 \leq i \leq m}$ verifies: $y_i = g(x_i) + \epsilon$ where ϵ is independent and identically distributed (i.i.d.) zero mean random error. Although we provide a sample set with additive noise, the objective is to learn the function g . In our coevolutionary processes for finding the function, the GP will try to guess the shape of the function whereas the EA will try to adjust the coefficients of the function. The motivation is to exploit the following intuitive idea: once the shape of the symbolic expression representing some optimal function has been found, we try to determine the best values of the coefficients appearing in the symbolic expression. One simple way to exemplify this situation is the following. Assume that we have to guess the equation of a geometric figure. If somebody (for example a GP algorithm) tells us that this figure is a quartic

function, it only remains for us to guess the appropriate coefficients. This point of view is not new and it constitutes the underlying idea of many successful methods in Machine Learning that combine a space of hypotheses with least square methods. Previous work in which constants of a symbolic expression have been effectively optimized has also dealt with memetic algorithms, in which classical local optimization techniques as gradient descent [8], linear scaling [9] or other methods based on diversity measures [10] were used.

The paper is organized as follows: section 2 provides the definition of the structure that will represent the programs and also includes the details of the designed GP algorithm. In section 3 we describe the EA for obtaining good values for the constants. Section 4 presents the cooperative coevolutionary architecture used for solving symbolic regression problem instances. In section 5 an experimental comparative study of the performance of our coevolutionary strategies is done. Finally, section 6 draws some conclusions.

2 GP with Straight Line Programs

In the GP paradigm, the evolved computer programs are usually represented by directed trees with ordered branches [11]. We use in this paper a structure for representing symbolic expressions by means of programs with straight line code. This structure is called *straight line program* (SLP). A SLP consists of a finite sequence of computational assignments where each assignment is obtained by applying some function to a set of arguments that can be variables, constants or pre-computed results. The SLP structure can describe complex computable functions using less amount of computational resources than GP-trees, as they can reuse previously computed results during the evaluation process. Now follows the formal definition of this structure.

Definition 1. Let $F = \{f_1, \dots, f_n\}$ be a set of functions, where each f_i has arity a_i , $1 \leq i \leq n$, and let $T = \{t_1, \dots, t_m\}$ be a set of terminals. A straight line program (SLP) over F and T is a finite sequence of computational instructions $\Gamma = \{I_1, \dots, I_l\}$, where for each $k \in \{1, \dots, l\}$, $I_k \equiv u_k := f_{j_k}(\alpha_1, \dots, \alpha_{a_{j_k}})$; with $f_{j_k} \in F$, $\alpha_i \in T$ for all i if $k = 1$ and $\alpha_i \in T \cup \{u_1, \dots, u_{k-1}\}$ for $1 < k \leq l$.

The set of terminals T satisfies $T = V \cup C$ where $V = \{x_1, \dots, x_p\}$ is a finite set of variables and $C = \{c_1, \dots, c_q\}$ is a finite set of constants. The number of instructions l is the length of Γ .

Observe that a SLP $\Gamma = \{I_1, \dots, I_l\}$ is identified with the set of variables u_i that are introduced by means of the instructions I_i . Thus the SLP Γ can be denoted by $\Gamma = \{u_1, \dots, u_l\}$. Each of the non-terminal variables u_i represents an expression over the set of terminals T constructed by a sequence of recursive compositions from the set of functions F .

An output set of a SLP $\Gamma = \{u_1, \dots, u_l\}$ is any set of non-terminal variables of Γ , that is $O(\Gamma) = \{u_{i_1}, \dots, u_{i_t}\}$. Provided that $V = \{x_1, \dots, x_p\} \subset T$ is the set of terminal variables, the function computed by Γ , denoted by $\Phi_\Gamma : I^p \rightarrow O^t$, is defined recursively in the natural way and satisfies $\Phi_\Gamma(a_1, \dots, a_p) = (b_1, \dots, b_t)$, where b_j stands for the value of the expression over V of the non-terminal variable u_{i_j} when we substitute the variable x_k by a_k ; $1 \leq k \leq p$.

Example 1. Let F be the set given by the three binary standard arithmetic operations, $F = \{+, -, *\}$ and let $T = \{1, x_1, x_2\}$ be the set of terminals. In this situation any SLP over F and T is a finite sequence of instructions where each instruction represents a polynomial in two variables with integer coefficients. If we consider the following SLP Γ of length 5 with output set $O(\Gamma) = \{u_5\}$:

$$\Gamma \equiv \begin{cases} u_1 := x_1 + 1 \\ u_2 := u_1 * u_1 \\ u_3 := x_2 + x_2 \\ u_4 := u_2 * u_3 \\ u_5 := u_4 - u_3 \end{cases} \quad (2)$$

the function computed by Γ is the polynomial

$$\Phi_\Gamma = 2x_2(x_1 + 1)^2 - 2x_2$$

Straight line programs have a large history in the field of Computational Algebra. A particular class of straight line programs, known in the literature as arithmetic circuits, constitutes the underlying computation model in Algebraic Complexity Theory [12]. They have been used in linear algebra problems [13], in quantifier elimination [14] and in algebraic geometry [15]. Recently, SLP's have been presented as a promising alternative to the trees in the field of Genetic Programming, with a good performance in solving some regression problem instances [16]. A SLP $\Gamma = \{u_1, \dots, u_l\}$ over F and T with output set $O(\Gamma) = \{u_l\}$ could also be considered as a grammar with $T \cup F$ as the set of terminals, $\{u_1, \dots, u_l\}$ as the set of variables, u_1 the start variable and the instructions of Γ as the rules. This grammar only generates one word that is the expression represented by the slp Γ . Note that this is not exactly Grammar Evolution (GE). In GE there is a user specified grammar and the individuals are integer strings which code for selecting rules from the provided grammar. In our case each individual is a context-free grammar generating a context-free language of size one.

Hence we will work with SLP's over a set F of functions and a set T of terminals. The elements of T that are constants, i.e. $C = \{c_1, \dots, c_q\}$, they are not fixed numeric values but references to numeric values. Hence, by specializing each c_i to a fixed value we obtain a specific SLP whose corresponding semantic function is a candidate solution for the problem instance.

For constructing each individual Γ of the initial population, we adopt the following process: for each instruction $u_k \in \Gamma$ first an element $f \in F$ is randomly selected and then its arguments are also randomly chosen in $T \cup \{u_1, \dots, u_{k-1}\}$ if $k > 1$ and in T if $k = 1$. We will consider populations with individuals of equal length L , where L is selected by the user. In this sense, note that given a SLP $\Gamma = \{u_1, \dots, u_l\}$ and $L \geq l$, we can construct the SLP $\Gamma' = \{u_1, \dots, u_{l-1}, u'_l, \dots, u'_{L-1}, u'_L\}$, where $u'_L = u_l$ and u'_k , for $k \in \{l, \dots, L-1\}$, is any instruction. If we consider the same output set for Γ and Γ' is easy to see that they represent the same function, i.e. $\Phi_\Gamma \equiv \Phi_{\Gamma'}$.

Assume a symbolic regression problem instance with a sample set $z = (x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}$, $1 \leq i \leq m$, and let Γ be a specific SLP over F and T obtained by means of the specialization of the constant references $C = \{c_1, \dots, c_q\}$. In this situation, the

empirical error of Γ with respect to the sample set of data points z is defined by the following expression:

$$\varepsilon_z(\Phi_\Gamma) = \frac{1}{m} \sum_{i=1}^m (\Phi_\Gamma(x_i) - y_i)^2 \quad (3)$$

We will use the following recombination operators for the SLP structure.

2.1 SLP-Crossover

Let $\Gamma = \{u_1, \dots, u_L\}$ and $\Gamma' = \{u'_1, \dots, u'_L\}$ be two SLP's over F and T . For the construction of an offspring, first a position k in Γ is randomly selected; $1 \leq k \leq L$. Let $S_{u_k} = \{u_{j_1}, \dots, u_{j_m}\}$ be the set of instructions of Γ involved in the evaluation of u_k . Assume that $j_1 < \dots < j_m$. Next we randomly select a position t in Γ' with $m \leq t \leq L$ and we substitute in Γ' the subset of instructions $\{u'_{t-m+1}, \dots, u'_t\}$ by the instructions of Γ in S_{u_k} suitably renamed. The renaming function \mathcal{R} applied to the elements of S_{u_k} is defined as $\mathcal{R}(u_{j_i}) = u'_{t-m+i}$, for all $i \in \{1, \dots, m\}$. With this process we obtain the first offspring of the crossover operation. For the second offspring we analogously repeat this strategy, but now selecting first a position k' in Γ' .

The underlying idea of the SLP-crossover is to interchange subexpressions between Γ and Γ' . The following example illustrates this fact.

Example 2. Let us consider two SLP's:

$$\Gamma \equiv \begin{cases} \mathbf{u}_1 := \mathbf{x} + \mathbf{y} \\ u_2 := u_1 * u_1 \\ \mathbf{u}_3 := \mathbf{u}_1 * \mathbf{x} \\ u_4 := u_3 + u_2 \\ u_5 := u_3 * u_2 \end{cases} \quad \Gamma' \equiv \begin{cases} \mathbf{u}_1 := \mathbf{x} * \mathbf{x} \\ \mathbf{u}_2 := \mathbf{u}_1 + \mathbf{y} \\ u_3 := u_1 + x \\ \mathbf{u}_4 := \mathbf{u}_2 * \mathbf{x} \\ u_5 := u_1 + u_4 \end{cases}$$

If $k = 3$ then $S_{u_3} = \{u_1, u_3\}$ (in bold font). t must be selected in $\{2, \dots, 5\}$. Assumed that $t = 3$, then the first offspring is:

$$\Gamma_1 \equiv \begin{cases} u_1 := x * x \\ \mathbf{u}_2 := \mathbf{x} + \mathbf{y} \\ \mathbf{u}_3 := \mathbf{u}_2 * \mathbf{x} \\ u_4 := u_2 * x \\ u_5 := u_1 + u_4 \end{cases}$$

that contains the subexpression of Γ represented by u_3 , and the rest of its instructions are taken from Γ' . For the second offspring, if the selected position in Γ' is $k' = 4$, then $S_{u_4} = \{u_1, u_2, u_4\}$. Now if $t' = 5$, the second offspring is:

$$\Gamma_2 \equiv \begin{cases} u_1 := x + y \\ u_2 := u_1 * u_1 \\ \mathbf{u}_3 := \mathbf{x} * \mathbf{x} \\ \mathbf{u}_4 := \mathbf{u}_3 + \mathbf{y} \\ \mathbf{u}_5 := \mathbf{u}_4 * \mathbf{x} \end{cases}$$

2.2 Mutation

When mutation is applied to a slp T , the first step consists in selecting an instruction $u_i \in T$ at random. Then a new random selection is made within the arguments of the function $f \in F$ that appears in the instruction u_i . Finally the selected argument is substituted by another one in $T \cup \{u_1, \dots, u_{i-1}\}$ randomly chosen.

As it is well known, the reproduction operation applied to an individual returns an exact copy of the individual.

2.3 Fitness Functions

In general, the real error of a model f as a solution of some symbolic regression problem is written as:

$$\varepsilon(f) = \int Q(x, f; y) d\mu, \quad (4)$$

where Q measures some notion of loss between $f(x)$ and y , and μ is the distribution from which examples are obtained. In our case we take $Q(x, f; y) = (f(x) - y)^2$. Nevertheless, the starting point of Statistical Learning Theory is that we might not know μ . At this point one replace the theoretical real error $\varepsilon(f)$ by the empirical error that is estimated from the finite sample set (equation 1); and GP procedure can use as fitness function the empirical error to find the model that minimizes it.

However, given a sample data set obtained by means of the observation, it could seem that the best model might fit exactly the data, but this situation would lead to a poor performance on unseen instances and in the presence of noise. Hence, the general idea is to look for a model as simple as possible that fits well the data set. This raises the question of how to measure the complexity of a model. There is no universal way to measure the complexity of the model and the choice of a specific measure depends on the problem at hand. Usually, for tree structures, a measure for the complexity is the height or the width of the tree. For our structure of SLP that represents the model and motivated by the concept of degree for the case of polynomials, we consider the number of non-scalar instructions in order to measure the complexity. The non-scalar instructions are those in which the selected operator in F is different from $\{+, -\}$.

Analytic model selection criteria estimate the real error displayed in equation 4 as a function of the empirical error with a penalty term related with the measure of the model complexity. This function usually takes the following form:

$$\varepsilon(f) \simeq \varepsilon_m(f) * pen(h, m); \quad * \in \{+, \cdot\} \quad (5)$$

where f is the model, h is the model complexity and m is the size of the sample set. Formally, the above expression arises from a previous generalization error bound of the same type. The results by Vapnik ([17]) state the following error bound that does not depend on the distribution μ .

$$\varepsilon(f) \leq \varepsilon_m(f) + \sqrt{\frac{h(\log(2m/h) + 1) - \log(\eta/4)}{m}} \quad (6)$$

where η is the probability that bound is violated and the considered complexity measure h is the so called Vapnik-Chervonenkis (VC) dimension. This penalty term is known as the VC confidence.

The Vapnik-Chervonenkis dimension (VC-dimension) of a family \mathcal{H} of models, functions or learning machines, represented by a class of programs, is a measure of the capacity of the family as classifier. In a binary classification problem, an instance x is classified by a label $y \in \{-1, 1\}$. Given a vector of n instances, (x_1, \dots, x_n) there are 2^n possible classification tuples (y_1, \dots, y_n) , with $y_i \in \{-1, 1\}$. If for each classification tuple (y_1, \dots, y_n) there is a classifier $f \in \mathcal{H}$ with $f(x_i) = y_i$, for $1 \leq i \leq n$, we say that (x_1, \dots, x_n) is shattered by the family \mathcal{H} . The VC-dimension of \mathcal{H} is defined as the maximum number n of points that can be shattered by \mathcal{H} . Thus, if the VC-dimension is h this means that there exists at least some set of h points which can be shattered. For instance, in binary classification, the VC-dimension of lines in the plane is 3, because there exists a set of 3 points in the plane such that for each classification tuple we can draw a line that keeps the points classified by -1 on one side and the points classified by 1 on the other side. This is not possible for any set of 4 points.

Some results about the relationship between the VC-dimension of a family of SLP's and the number of non-scalar instructions can be found in [18] or [19]. In all cases there is an upper bound of the VC-dimension that is polynomial in the number of non-scalar instructions.

In our case, for practical use of equation 6 we adopt the following formula as a new fitness function (see [20] or [17] for the derivation of this formula):

$$\varepsilon_m(f) \left(1 - \sqrt{p - p \ln p + \frac{\ln m}{2m}} \right)^{-1} \quad (7)$$

In the above equation, $p = \frac{h}{m}$ and h stands for the VC-dimension of the family \mathcal{H} that contains f . As an estimator of the VC-dimension h we will consider the number of non-scalar instructions of the SLP Γ that represents f (i.e. $f \equiv \Phi_\Gamma$)

3 The EA to Adjust the Constants

In this section we describe an EA that provides good values for the numeric terminal symbols $C = \{c_1, \dots, c_q\}$ appearing in the populations of SLP's that evolve during the GP process. Assume a population $P = \{\Gamma_1, \dots, \Gamma_N\}$ constituted by N SLP's over F and $T = V \cup C$. Let $[a, b] \subset \mathbb{R}$ be the search space for the constants c_i , $1 \leq i \leq q$. In this situation, an individual \bar{c} is represented by a vector of floating point numbers in $[a, b]^q$.

There are several ways of defining the fitness of a vector of constants \bar{c} , but in all of them the current population P of SLP's that evolves in the GP process is needed. So, given a sample set $z = (x_i, y_i) \in \mathbb{R}^n \times \mathbb{R}$, $1 \leq i \leq m$, that defines a symbolic regression instance, and given a vector of values for the constants $\bar{c} = (c_1, \dots, c_q)$, we could define the fitness of \bar{c} as follows:

$$\mathcal{F}_z^{EA}(\bar{c}) = \min\{\mathcal{F}_z(\Gamma_i^{\bar{c}}); 1 \leq i \leq N\} \quad (8)$$

where $\mathcal{F}_z(I_i^{\bar{c}})$ is computed by equation 7 and represents the fitness of the SLP I_i after the specialization of the references in C to the corresponding real values of \bar{c} .

Observe that when the fitness of \bar{c} is computed by means of the above formula, the GP fitness values of a whole population of SLP's are also computed. This could be a lot of computational effort when the size of both populations increases. In order to prevent the above situation new fitness functions for \bar{c} can be considered, where only a subset of the population P of the SLP's is evaluated. Previous work in cooperative coevolutionary architectures suggests two basic methods for selecting the subset of collaborators [5]. The first one in our case consists in the selection of the best SLP of the current population P corresponding the GP process. The second one selects two individuals from P : the best one and a random SLP.

3.1 Crossover

We will use arithmetic crossover [21]. Thus, in our EA, the crossover of two individuals \bar{c}_1 and $\bar{c}_2 \in [a, b]^q$ produces two offsprings \bar{c}'_1 and \bar{c}'_2 which are linear combinations of their parents.

$$\bar{c}'_1 = \lambda \cdot \bar{c}_1 + (1 - \lambda) \cdot \bar{c}_2; \quad \bar{c}'_2 = \lambda \cdot \bar{c}_2 + (1 - \lambda) \cdot \bar{c}_1 \quad (9)$$

In our implementation we randomly choose $\lambda \in (0, 1)$ for each crossover operation.

3.2 Mutation

A non-uniform mutation operator adapted to our search space $[a, b]^q$, which is convex, is used [22]. The following expressions define our mutation operator, with $p = 0.5$.

$$\bar{c}_k^{t+1} = \bar{c}_k^t + \Delta(t, b - \bar{c}_k^t), \text{ with probability } p \quad (10)$$

and

$$\bar{c}_k^{t+1} = \bar{c}_k^t - \Delta(t, \bar{c}_k^t - a), \text{ with probability } 1 - p \quad (11)$$

$k = 1, \dots, q$ and t is the current generation. The function Δ is defined as $\Delta(t, y) = y \cdot r \cdot (1 - \frac{t}{T})$ where r is a random number in $[0, 1]$ and T represents the maximum number of generations. Note that function $\Delta(t, y)$ returns a value in $[0, y]$ such that the probability of obtaining a value of $\Delta(t, y)$ close to zero increases as t increases. Hence the mutation operator searches the space uniformly initially (when t is small), and very locally at later stages.

In our EA we will use q -tournament as the selection procedure.

4 The Coevolutionary Architecture

In our case the EA for tuning the constants is subordinated to the main GP process with the SLP's. Hence, several collaborators are used during the computation of the fitness of a vector of constants \bar{c} , whereas only the best vector of constants is used to compute the fitness of a population of SLP's.

The cooperative coevolutionary strategy begins with the initialization of both populations. First the fitness of the individuals of the SLP's population are computed considering a randomly selected vector of constants as collaborator. Then alternative turns of both cooperative algorithms are performed. We will consider a *turn* as the isolated and uninterrupted evolution of one population for a fixed number of generations. We display below the algorithm describing this cooperative coevolutionary architecture:

```

begin
Pop_slp := initialize_GP-slp_population
Pop_const := initialize_EA-constants_population
Const_collabor := random(Pop_const)
evaluate(Pop_slp,Const_collabor)
While (not termination condition) do
  Pop_slp := turn_GP(Pop_slp,Const_collabor)
  Collabor_slp := {best(Pop_slp),
                  random(Pop_slp)}
  Pop_const := turn_EA(Pop_const,Collabor_slp)
  Const_collabor := best(Pop_const)
end

```

5 Experimentation

5.1 Experimental Settings

The experimentation consists in the execution of the proposed cooperative coevolutionary strategies, considering several types of target functions. We have corrupted the sample set by additive gaussian noise of level 1. Two experiments were performed.

For the first experiment two groups of target functions are considered: the first group includes 300 randomly generated univariate polynomials whose degrees are bounded by 5 and the second group consists of 300 target functions represented by randomly generated SLP's over $F = \{+, -, *, /, sqrt, sin, cos, ln, exp\}$ and $T = \{x, c\}, c \in [-1, 1]$, with length 16. We will name this second group "target SLP's".

A second experiment is also performed solving symbolic regression problem instances associated to the following two multivariate functions:

$$f_1(x, y, z) = (x + y + z)^2 + 1 \quad (12)$$

$$f_2(x, y) = x y + sin((x - 1)(y + 1)) \quad (13)$$

For every execution the noisy sample set is constituted by 30 points. In the case of the functions that belong to the first experiment, the sample points are in the range $[-1, 1]$. For the function f_1 the points are in the range $[-100, 100]$ for all variables. Finally function f_2 varies in the range $[-3, 3]$ along each axis.

The individuals are SLP's over $F = \{+, -, *, /, sqrt\}$ in the executions related to the 300 generated polynomials and to the function f_1 . The function set F is incremented with the operation sin for the problem instance associated to f_2 and also with the operations cos , ln and exp for the group of target SLP's.

Besides the variables, the terminal set also includes two references to constants for the polynomials and only one reference to a constant for the rest of the target functions. The constants take values in $[-1, 1]$.

The particular settings for the parameters of the GP process are the following: population size: 200, crossover rate: 0.9, mutation rate: 0.05, reproduction rate: 0.05, 2-tournament as selection procedure and maximum length of the SLP's: 16. In the case of the EA that adjusts the constants, the population includes 100 vector of constants, crossover rate: 0.9, mutation rate: 0.1 and also 2-tournament as selection procedure. For all the coevolutionary strategies, the computation of the fitness of an SLP during the GP process will use the best vector of constants as collaborator, whereas in order to compute the fitness of a vector of constants in the EA process, we consider a collaborator set containing the best SLP of the population and another one randomly selected. Both processes are elitist and a generational replacement between populations is used. But in the construction of the new population, the offsprings generated do not necessarily replace their parents. After a crossover we have four individuals: two parents and two offsprings. We select the two best individuals with different fitness values. Our motivation is to prevent premature convergence and to maintain diversity in the population.

We compare the standard GP-slp strategy without coevolution with two coevolutionary strategies that follow the general architecture described in section 4. We consider two fitness functions for the populations of SLP's: The empirical error \mathcal{F}_1 defined by equation 1 and the fitness with a penalty term \mathcal{F}_2 described by equation 7.

The first coevolutionary strategy, named Turns GP-EA (TGPEA), consists in the execution of alternative turns of each cooperative algorithm. The second strategy executes first a large turn of the GP algorithm with the SLP's and then follows the execution of the EA related with the constants until termination condition was reached. This strategy is named Separated GP-EA (SGPEA). In the case of TGPEA we have considered a GP turn as the evolution of the population of SLP's during 25 generations. On the other hand, an EA turn consists of 5 generations in the evolution of the population related to the constants. In SGPEA strategy we divide the computational effort between the two algorithms: 90% for GP and 10% for EA. The computational effort (CE) is defined as the total number of basic operations that have been computed up to that moment. We will denote the use of each fitness function, adding the suffix \mathcal{F}_1 or \mathcal{F}_2 to the name of the corresponding strategy.

In the first experiment one execution for each strategy has been performed over the 600 generated target functions. On the other hand, in the second experiment we have executed all strategies 300 times for each of the two multivariate functions f_1 , and f_2 . For all the executions the evolution finished after 10^7 basic operations have been computed.

5.2 Experimental Results

Frequently, when different Genetic Programming strategies for solving symbolic regression instances are compared, the quality of the final selected model is evaluated by means of its corresponding fitness value over the sample set. But with this quality measure it is not possible to distinguish between good executions and overfitting executions. Then it makes sense to consider another new set of unseen points without noise, called the *validation set*, in order to give a more appropriate indicator of the quality of

the selected model. So, let $(x_i, y_i)_{1 \leq i \leq n_{test}}$ a validation set for the target function $g(x)$ (i.e. $y_i = g(x_i)$) and let $f(x)$ be the model estimated from the sample set. Then the *validation fitness* $vf_{n_{test}}$ is defined by the mean square error (MSE) between the values of f and the true values of the target function g over the validation set:

$$vf_{n_{test}} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (f(x_i) - y_i)^2 \tag{14}$$

An execution will be considered successful if the final selected model f has validation fitness less than 10% of the range of the sample set $z = (x_i, y_i)_{1 \leq i \leq 30}$. That is:

$$vf_{n_{test}} \leq 0.1 \left| \max_{1 \leq i \leq 30} y_i - \min_{1 \leq i \leq 30} y_i \right| \tag{15}$$

On the other hand, an execution will be spurious if the validation fitness of the selected model verifies:

$$vf_{n_{test}} \geq 1.5 |Q_3 - Q_1| \tag{16}$$

Were Q_1 and Q_3 represent, respectively, the first and third quartile of the empirical distribution of the executions in terms of the validation fitness. The spurious executions will be removed from the experiment.

In what follows we shall present a statistical comparative study about the performance of the described coevolutionary strategies. For both experiments we will show the empirical distribution of the non-spurious executions as well as the values of the mean, variance, median, worst and best execution in terms of the validation fitness. We also present statistical hypothesis tests in order to determine if some strategy is better than the others. We consider a validation set of 200 new and unseen points randomly generated.

Experiment 1. We shall denote the polynomial set as $P_5^R[X]$ and the set of target SLP's over F and T as $SLP(F, T)$. Table 1 displays for each strategy the spurious and success rates of the executions. Note that the success rate is computed after removing the spurious executions.

Table 1. Spurious and success rates for each strategy and group of target functions

	$P_5^R[X]$		$SLP(F, T)$	
	spurious	success	spurious	success
$TGPEA_{\mathcal{F}_1}$	13%	6%	20%	13%
$SGPEA_{\mathcal{F}_1}$	13%	5%	20%	17%
$GP - slp_{\mathcal{F}_1}$	13%	4%	22%	15%
$TGPEA_{\mathcal{F}_2}$	7%	3%	19%	13%
$SGPEA_{\mathcal{F}_2}$	5%	4%	20%	10%
$GP - slp_{\mathcal{F}_2}$	7%	2%	20%	11%

Figure 1 presents the empirical distribution of the executions over the two groups of generated target functions. This empirical distribution is displayed using standard box plot notation with marks at best execution, 25%, 50%, 75% and worst execution,

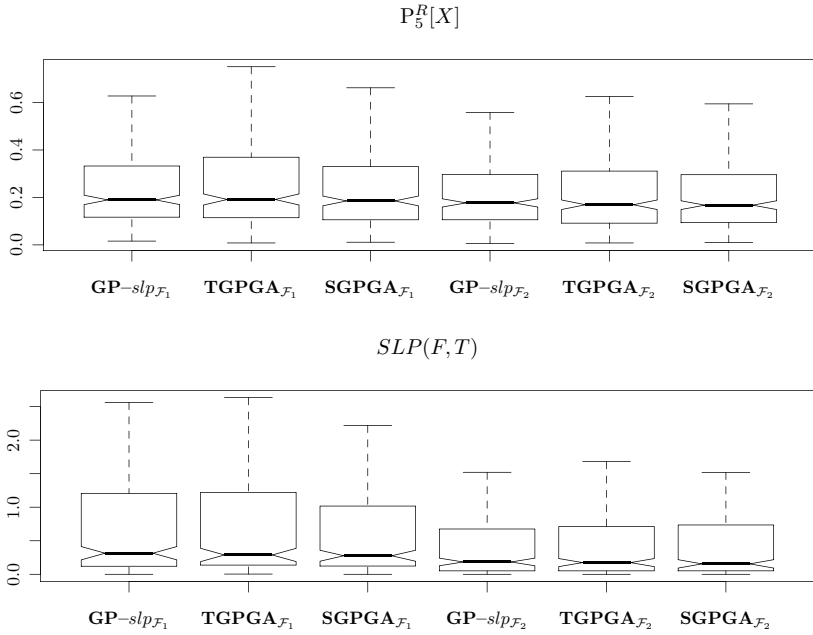


Fig. 1. Empirical distributions of the non-spurious executions

considering the validation fitness of the selected model. Table 2 specifies the values of the validation fitness for the worst, median and best execution, as well as the means and variances. Note that for these two groups of functions one execution per target function was performed.

Table 2. Minimal, median and maximal values of the validation fitness for each method and group of target function. Also values of means and variances are showed.

$P_5^R[X]$	min	med	max
$TGPEA_{\mathcal{F}_1}$	$8.16 \cdot 10^{-3}$	0.17	0.7
$SGPEA_{\mathcal{F}_1}$	$1.11 \cdot 10^{-2}$	0.17	0.66
$GP - slp_{\mathcal{F}_1}$	$1.6 \cdot 10^{-2}$	0.16	0.63
$TGPEA_{\mathcal{F}_2}$	$8.18 \cdot 10^{-3}$	0.15	0.63
$SGPEA_{\mathcal{F}_2}$	$9.87 \cdot 10^{-3}$	0.16	0.59
$GP - slp_{\mathcal{F}_2}$	$6.35 \cdot 10^{-3}$	0.16	0.56
$SLP(F, T)$	min	med	max
$TGPEA_{\mathcal{F}_1}$	$3.44 \cdot 10^{-3}$	0.23	2.63
$SGPEA_{\mathcal{F}_1}$	0	0.19	2.22
$GP - slp_{\mathcal{F}_1}$	0	0.21	2.56
$TGPEA_{\mathcal{F}_2}$	0	0.11	1.47
$SGPEA_{\mathcal{F}_2}$	0	$9.29 \cdot 10^{-2}$	1.52
$GP - slp_{\mathcal{F}_2}$	0	0.11	1.52

$P_5^R[X]$	μ	σ
$TGPEA_{\mathcal{F}_1}$	0.21	0.14
$SGPEA_{\mathcal{F}_1}$	0.19	0.13
$GP - slp_{\mathcal{F}_1}$	0.19	0.12
$TGPEA_{\mathcal{F}_2}$	0.19	0.13
$SGPEA_{\mathcal{F}_2}$	0.19	0.12
$GP - slp_{\mathcal{F}_2}$	0.19	0.13
$SLP(F, T)$	μ	σ
$TGPEA_{\mathcal{F}_1}$	0.37	0.44
$SGPEA_{\mathcal{F}_1}$	0.32	0.35
$GP - slp_{\mathcal{F}_1}$	0.34	0.43
$TGPEA_{\mathcal{F}_2}$	0.23	0.3
$SGPEA_{\mathcal{F}_2}$	0.21	0.29
$GP - slp_{\mathcal{F}_2}$	0.22	0.27

Analyzing the information given by the above tables and figure we could deduce the following facts:

1. The penalized fitness \mathcal{F}_2 seems to produce less spurious executions but also less success executions. This is more clear for the case of the polynomials. Hence we can say that \mathcal{F}_2 produces executions more homogeneous. On the other hand, the success rates are low because we have corrupted the sample set with a noise level of 1. Considering noise levels of 0.1 and 0.2, the success rates are for all cases above 75%.
2. For the group of polynomials, the empirical distributions of the non-spurious runs are very similar for all the studied strategies. Nevertheless, for the target SLP's, probably the coevolutionary strategies with the penalized fitness are slightly better than the others. Observe in figure 1 that these strategies have the corresponding boxes smaller and a little below than the other methods.
3. The values displayed in table 2 permit to obtain the same conclusions as those presented above. However all strategies perform quite well over the target functions of this experiment.

With the objective of justify the comparative quality of the studied strategies we have made statistical hypothesis tests between them, which results are showed in table 3. Roughly speaking, the null-hypothesis in each test with associated pair (i, j) is that strategy i is not better than strategy j . Hence if value a_{ij} of the element (i, j) in table 3 is less than a significance value α , we can reject the corresponding null-hypothesis.

From the results presented in table 3 and with a significance value of $\alpha = 0.05$, we can conclude that for the group of polynomials there is not a winner strategy whereas for the group of target SLP's the fitness \mathcal{F}_2 is clearly better than the empirical error fitness \mathcal{F}_1 .

Table 3. Results of the crossed statistical hypothesis tests about the comparative quality of the studied strategies

$P_5^R[X]$	$TGPEA_{\mathcal{F}_1}$	$SGPEA_{\mathcal{F}_1}$	$GP - slp_{\mathcal{F}_1}$	$TGPEA_{\mathcal{F}_2}$	$SGPEA_{\mathcal{F}_2}$	$GP - slp_{\mathcal{F}_2}$
$TGPEA_{\mathcal{F}_1}$	1	0.91	0.63	0.99	0.76	0.95
$SGPEA_{\mathcal{F}_1}$	0.32	1	0.33	0.66	0.84	0.64
$GP - slp_{\mathcal{F}_1}$	0.34	0.83	1	0.66	0.81	0.63
$TGPEA_{\mathcal{F}_2}$	$2.58 \cdot 10^{-2}$	0.21	0.12	1	0.47	0.37
$SGPEA_{\mathcal{F}_2}$	0.15	0.48	0.22	0.77	1	0.41
$GP - slp_{\mathcal{F}_2}$	0.35	0.73	0.42	0.72	0.62	1
$SLP(F, T)$	$TGPEA_{\mathcal{F}_1}$	$SGPEA_{\mathcal{F}_1}$	$GP - slp_{\mathcal{F}_1}$	$TGPEA_{\mathcal{F}_2}$	$SGPEA_{\mathcal{F}_2}$	$GP - slp_{\mathcal{F}_2}$
$TGPEA_{\mathcal{F}_1}$	1	0.83	0.62	1	1	1
$SGPEA_{\mathcal{F}_1}$	0.17	1	0.6	0.99	1	1
$GP - slp_{\mathcal{F}_1}$	0.15	0.58	1	1	1	1
$TGPEA_{\mathcal{F}_2}$	$3.69 \cdot 10^{-10}$	$2.14 \cdot 10^{-7}$	$7.73 \cdot 10^{-7}$	1	0.95	0.61
$SGPEA_{\mathcal{F}_2}$	$1.85 \cdot 10^{-12}$	$1.48 \cdot 10^{-9}$	$6.98 \cdot 10^{-8}$	0.39	1	0.39
$GP - slp_{\mathcal{F}_2}$	$9.68 \cdot 10^{-10}$	$5.33 \cdot 10^{-7}$	$1.25 \cdot 10^{-6}$	0.15	0.71	1

Experiment 2. In this experiment, the multivariate functions described by the expressions [I2](#) and [I3](#) have been considered as target functions. We have performed 300 executions for each strategy and function. In the following tables and figures we present for the new functions the same results as those presented for the target functions of experiment 1.

In terms of success rates, all methods are of similar performance. The *SGPEA* strategy, with both fitness functions, seems to be better than the others. Observing figure [2](#) we can confirm that for the target function f_1 the separated method outperforms the others. Surprisingly, the penalized fitness did not perform very well for the pure GP strategy nor for the alternative turns strategy. On the other hand, for the target function f_2 is not clear that the penalized fitness is the best one.

Table 4. Spurious and success rates for each strategy and target function

	$f_1(x, y, z)$		$f_2(x, y)$	
	spurious	success	spurious	success
$TGPEA_{\mathcal{F}_1}$	23%	87%	9%	100%
$SGPEA_{\mathcal{F}_1}$	17%	87%	9%	100%
$GP - slp_{\mathcal{F}_1}$	25%	86%	11%	100%
$TGPEA_{\mathcal{F}_2}$	24%	84%	12%	100%
$SGPEA_{\mathcal{F}_2}$	19%	85%	6%	100%
$GP - slp_{\mathcal{F}_2}$	23%	84%	10%	100%

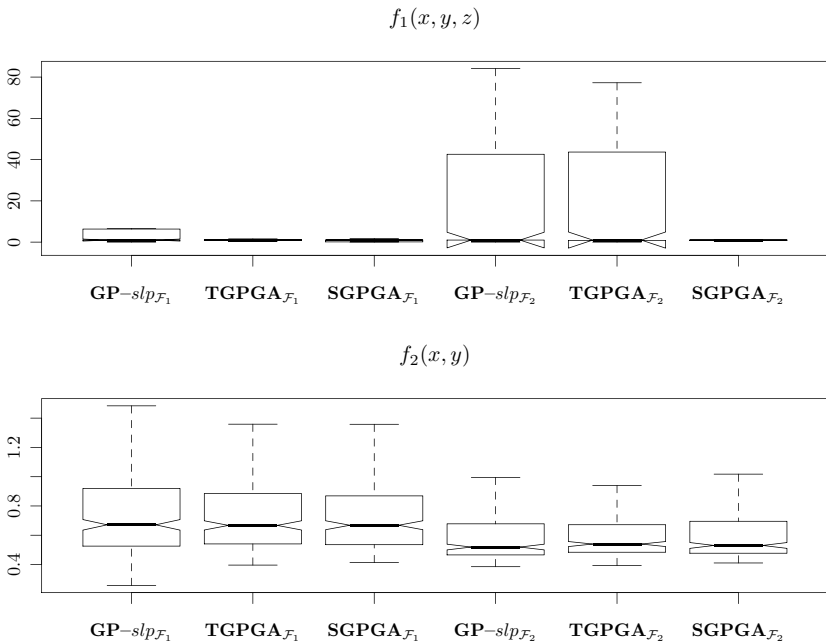


Fig. 2. Empirical distributions of the non-spurious executions

Table 5. Minimal, median and maximal values of the validation fitness for each method and target function. A table with means and variances is also displayed.

$f_1(x, y, z)$	min	med	max	$f_1(x, y, z)$	μ	σ
$TGPEA_{\mathcal{F}_1}$	$1.51 \cdot 10^{-6}$	1	1.58	$TGPEA_{\mathcal{F}_1}$	0.76	0.42
$SGPEA_{\mathcal{F}_1}$	$3.6 \cdot 10^{-6}$	1	1.78	$SGPEA_{\mathcal{F}_1}$	0.7	0.48
$GP - slp_{\mathcal{F}_1}$	$2.48 \cdot 10^{-5}$	1	6,64	$GP - slp_{\mathcal{F}_1}$	0.91	0.84
$TGPEA_{\mathcal{F}_2}$	$1.54 \cdot 10^{-5}$	1	77.25	$TGPEA_{\mathcal{F}_2}$	1.7	6.91
$SGPEA_{\mathcal{F}_2}$	$9.67 \cdot 10^{-8}$	1	1.18	$SGPEA_{\mathcal{F}_2}$	0.71	0.45
$GP - slp_{\mathcal{F}_2}$	$2.71 \cdot 10^{-6}$	1	84.16	$GP - slp_{\mathcal{F}_2}$	2.79	10.83
$f_2(x, y)$	min	med	max	$f_2(x, y)$	μ	σ
$TGPEA_{\mathcal{F}_1}$	0.4	0.65	1.36	$TGPEA_{\mathcal{F}_1}$	0.69	0.21
$SGPEA_{\mathcal{F}_1}$	0.41	0.65	1.36	$SGPEA_{\mathcal{F}_1}$	0.69	0.21
$GP - slp_{\mathcal{F}_1}$	0.26	0.63	1.48	$GP - slp_{\mathcal{F}_1}$	0.7	0.22
$TGPEA_{\mathcal{F}_2}$	0.39	0.52	0.94	$TGPEA_{\mathcal{F}_2}$	0.56	0.12
$SGPEA_{\mathcal{F}_2}$	0.41	0.52	1.02	$SGPEA_{\mathcal{F}_2}$	0.58	0.14
$GP - slp_{\mathcal{F}_2}$	0.39	0.5	0.98	$GP - slp_{\mathcal{F}_2}$	0.56	0.13

Finally, as it was done in experiment 1, we have made for the three functions the crossed statistical hypothesis tests between all pairs of the considered strategies and the results are showed in table 6. Considering a significant value $\alpha = 0.05$, SGPEA is confirmed as the best strategy for target function f_1 and fitness \mathcal{F}_2 is clearly better for target function f_2 .

Table 6. Results of the crossed statistical hypothesis tests

$f_1(x, y, z)$	$TGPEA_{\mathcal{F}_1}$	$SGPEA_{\mathcal{F}_1}$	$GP - slp_{\mathcal{F}_1}$	$TGPEA_{\mathcal{F}_2}$	$SGPEA_{\mathcal{F}_2}$	$GP - slp_{\mathcal{F}_2}$
$TGPEA_{\mathcal{F}_1}$	1	0.88	0.25	0.26	0.52	0.36
$SGPEA_{\mathcal{F}_1}$	$2.54 \cdot 10^{-2}$	1	$5.81 \cdot 10^{-3}$	0.32	0.47	$1.45 \cdot 10^{-2}$
$GP - slp_{\mathcal{F}_1}$	0.26	0.44	1	$2.54 \cdot 10^{-2}$	1	0.71
$TGPEA_{\mathcal{F}_2}$	0.45	0.99	0.21	1	0.46	0.28
$SGPEA_{\mathcal{F}_2}$	$8.16 \cdot 10^{-2}$	0.26	$1.4 \cdot 10^{-2}$	$2.03 \cdot 10^{-2}$	1	$1.78 \cdot 10^{-2}$
$GP - slp_{\mathcal{F}_2}$	0.12	0.2	0.72	$5.37 \cdot 10^{-3}$	0.86	1
$f_2(x, y)$	$TGPEA_{\mathcal{F}_1}$	$SGPEA_{\mathcal{F}_1}$	$GP - slp_{\mathcal{F}_1}$	$TGPEA_{\mathcal{F}_2}$	$SGPEA_{\mathcal{F}_2}$	$GP - slp_{\mathcal{F}_2}$
$TGPEA_{\mathcal{F}_1}$	1	0.89	0.53	1	1	1
$SGPEA_{\mathcal{F}_1}$	0.72	1	0.51	0.99	1	1
$GP - slp_{\mathcal{F}_1}$	0.49	0.57	1	0.99	1	1
$TGPEA_{\mathcal{F}_2}$	$4.5 \cdot 10^{-15}$	$2.27 \cdot 10^{-14}$	$3.91 \cdot 10^{-12}$	1	0.18	0.62
$SGPEA_{\mathcal{F}_2}$	$6.71 \cdot 10^{-12}$	$1.64 \cdot 10^{-11}$	$2.43 \cdot 10^{-10}$	0.53	1	0.98
$GP - slp_{\mathcal{F}_2}$	$1.08 \cdot 10^{-14}$	$3.27 \cdot 10^{-14}$	$8.22 \cdot 10^{-13}$	$4.65 \cdot 10^{-3}$	$1.84 \cdot 10^{-2}$	1

6 Conclusions

We have designed two cooperative coevolutionary strategies between a GP and an EA. The genetic program evolves straight line programs that represent symbolic expressions whereas the evolutionary algorithm optimizes the values of the constants used by those

expressions. For the evaluation of the populations of SLP's we have considered the traditional fitness function based on the empirical error and another one with a penalty term that involves the number of non-scalar instructions of the corresponding SLP. Experimentation has been performed on several groups of target functions and we have compared the performance between the studied strategies. The quality of the selected model after the execution was measured considering a validation set of unseen points randomly generated, instead of the sample set used for the evolution process. A statistical study of the experimental results has been done. It has been shown that the straight line program is a good structure to represent symbolic expressions. Also we can conclude that the penalized fitness is the best option when the sample set is corrupted by noise.

Acknowledgements. This work was partially supported by the R&D program of the Community of Madrid (S2009/TIC-1650, project e-Madrid).

References

1. Maynard, J.: *Evolution and the theory of games*. Cambridge University Press, Cambridge (1982)
2. Axelrod, R.: *The evolution of cooperation*. Basic Books, New York (1984)
3. Hillis, D.: Co-evolving parasites improve simulated evolution as an optimization procedure. *Artificial Life II, SFI Studies in the Sciences Complexity* 10, 313–324 (1991)
4. Rosin, C., Belew, R.: New methods for competitive coevolution. *Evolutionary Computation* 5(1), 1–29 (1996)
5. Wiegand, R.P., Liles, W.C., De Jong, K.A.: An Empirical Analysis of Collaboration Methods in Cooperative Coevolutionary Algorithms. In: *Proceedings of the 2001 Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 1235–1242 (2001)
6. Casillas, J., Cordón, O., Herrera, F., Merelo, J.: Cooperative coevolution for learning fuzzy rule-based systems. In: *Genetic and Evolutionary Computation Conference (GECCO 2006)*, pp. 361–368 (2006)
7. Vanneschi, L., Mauri, G., Valsecchi, A., Cagnoni, S.: Heterogeneous Cooperative Coevolution: Strategies of Integration between GP and GA. In: *Proc. of the Fifth Conference on Artificial Evolution (AE 2001)*, pp. 311–322 (2001)
8. Topchy, A., Punch, W.F.: Faster genetic programming based on local gradient search of numeric leaf values. In: *Proceedings of the 2001 Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 155–162 (2001)
9. Keijzer, M.: Improving Symbolic Regression with Interval Arithmetic and Linear Scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) *EuroGP 2003*. LNCS, vol. 2610, pp. 71–83. Springer, Heidelberg (2003)
10. Ryan, C., Keijzer, M.: An Analysis of Diversity of Constants of Genetic Programming. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) *EuroGP 2003*. LNCS, vol. 2610, pp. 409–418. Springer, Heidelberg (2003)
11. Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge (1992)
12. Burguisser, P., Clausen, M., Shokrollahi, M.A.: *Algebraic Complexity Theory*. Springer, Heidelberg (1997)
13. Berkowitz, S.J.: On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters* 18, 147–150 (1984)

14. Heintz, J., Roy, M.F., Solerno, P.: Sur la complexite du principe de Tarski-Seidenberg. *Bulletin de la Societe Mathematique de France* 118, 101–126 (1990)
15. Giusti, M., Heintz, J., Morais, J., Morgenstern, J.E., Pardo, L.M.: Straight Line Programs in Geometric Elimination Theory. *Journal of Pure and Applied Algebra* 124, 121–146 (1997)
16. Alonso, C.L., Montana, J.L., Puente, J.: Straight line programs: a new Linear Genetic Programming Approach. In: *Proc. 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 517–524 (2008)
17. Vapnik, V.: *Statistical Learning Theory*. John Willey and Sons (1998)
18. Montaña, J.L., Alonso, C.L., Borges, C.E., Crespo, C.L.: Adaptation, Performance and Vapnik-Chervonenkis Dimension of Straight Line Programs. In: *Proc. 12th European Conference on Genetic Programming*, pp. 315–326 (2009)
19. Alonso, C.L., Montaña, J.L., Borges, C.E.: Model Complexity Control in Straight Line Program Genetic Programming. Technical Report (2011)
20. Cherkassky, V., Yunkian, M.: Comparison of Model Selection for Regression. *Neural Computation* 15(7), 1691–1714 (2003)
21. Schwefel, H.P.: *Numerical Optimization of Computer Models*. John Wiley and Sons, New-York (1981)
22. Michalewicz, Z., Logan, T., Swaminathan, S.: Evolutionary operators for continuous convex parameter spaces. In: *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pp. 84–97 (1994)

Evolution of Collective Perception in a Group of Autonomous Robots

Giuseppe Morlino¹, Vito Trianni², and Elio Tuci³

¹ ISTC-CNR, Rome, Italy

giuseppe.morlino@istc.cnr.it

² IRIDIA-CoDE, ULB, Brussels, Belgium

vtrianne@ulb.ac.be

³ Aberystwyth University, Aberystwyth, U.K.

elt7@aber.ac.uk

Abstract. In this paper, we present an evolutionary robotics experiment that aims at studying how a macroscopic variable can be encoded in the collective activity of a group of robots. In particular, we aim at understanding how perception can be the result of a collective, self-organising process. A group of robots is placed in an environment characterised by black spots painted on the ground. The density of the spots is the macroscopic variable that should be perceived by the group. The density varies from trial to trial, and robots are requested to collectively encode such density into a coherent signalling activity. Robots have access only to local information, therefore cannot immediately perceive the global density. By exploiting interactions through an all-to-all communication channel, robots should prove capable of perceiving and encoding the global density. We show how such behaviour can be synthesised exploiting evolutionary robotics techniques, and we present extensive analyses of the evolved strategies.

1 Introduction

How can a distributed system collectively encode the magnitude of a macroscopic variable? This question holds over multiple domains, and at different scales. First and foremost, in the context of cognitive neuroscience, this question can be reformulated as: what are the neural mechanisms underlying perception? This is a fundamental question, which must be answered first in order to lay the foundations for further investigations on other cognitive processes, such as decision-making, attention or learning. For this reason, the literature abounds of models about neural coding of every sort of stimuli, from the basic ones—e.g., vibro-tactile or visual stimuli [1,2]—to more complex perceptual conditions—e.g., multi-stability, face recognition or numbers [3,4,5,6].

The problem of suitably encoding environmental stimuli, however, does not pertain exclusively individual animals, but is of fundamental importance also for collective systems, such as bird flocks and honeybee swarms. Similar systems are often considered as super-organisms, due to their high cohesion and integrated functioning [7,8]. It is therefore interesting to look at how super-organisms can achieve a coherent perception of macroscopic features of the environment they inhabit. For instance, while searching a new nesting site, honeybees explore the environment thanks to scouts that report their

discoveries to the nest. In there, a collective perception and a decision-making process is carried on, which results in the recognition and selection of the best site among the discovered choices [9]. In this process, no single bee has the full picture. However, the partial information of many bees is aggregated in the nest and through a self-organising process decision-making is successfully performed. In [9], strong similarities are recognised between honeybees behaviour and the mechanisms that support perception and decision-making in neural systems. In particular, cross-inhibition within neural populations is functionally similar to negative feedback between bee workers committed to different nesting sites. The parallel between cognitive systems and swarm behaviour goes beyond qualitative considerations. In [10], the nest site selection behaviour in ants and honeybees is compared with the brain dynamics during decision making in a perceptual choice task. The author show that the swarm behaviour can be described by the same model that was proposed for decision making in [11]. As a consequence, the two decision processes can be directly compared, and similarities can be drawn between cognition in the brain and in the swarm.

In this paper, we aim at studying collective perception in a robotic swarm. The goal of this study is understanding which are the self-organising processes underlying the collective perception of a macroscopic environmental feature, which is not accessible to the individual robots due to their limited perceptual abilities and due to the nature of their individual exploration strategies. Therefore, multiple robots need to interact in order to give a collective response that correlates with the macroscopic variable. It is worth noticing that the perceptual discrimination task employed could in principle be solved by a single robot, given an effective exploration strategy and enough time to accomplish it. The reason why we let a group of robots to find a collective solution is because we believe that the study of successful collective discrimination strategies in this particular artificial scenario may shed a light on the mechanisms of collective perception in natural organisms.

In this robotic model, we synthesise the robot neural controllers through evolutionary robotic techniques, and we afterwards analyse the obtained results in order to uncover the mechanisms that support the collective perception process. Our working hypothesis is that the evolutionary process can produce optimal solutions to the given task. Therefore, by analysing these solutions, we can discover general mechanisms for collective perception, which are adapted to the experimental conditions we have devised. This allows us to discuss the discovered mechanisms with respect to known processes performed by individuals and collectives. The usage of evolutionary techniques for collective and swarm robotics has been demonstrated in various recent studies. For instance, in [12], self-organising synchronisation was evolved for a group of robots that presented an individual periodic behaviour. In [13], the behaviour for a robotic swarm was evolved through an evolutionary process in order to collectively explore the environment and form a path to navigate between two target areas, which were too distant to be perceived by a single agent at the same time. In [14], artificial evolution was exploited to synthesise Swarming Micro Air Vehicles (SMAVs) able to organise autonomously, relaying only on local informations, to establish a wireless communication network between users located on the ground.

The paper is organised as follows. In Section 2 we describe the experimental setup, giving details about the robot configuration, the characteristics of the environment, the controller and the evolutionary algorithm used to synthesise it. In Section 3, we describe the obtained results, and in particular we select and analyse two different controllers that are representative of two classes of evolved strategies. Finally, Section 4 concludes the paper with some discussion.

2 Experimental Setup

As mentioned above, in this paper we study how a swarm of robots can collectively encode a macroscopic feature of the environment. We have set up an experimental arena in which black circular spots are painted on a grey background. The macroscopic feature that must be encoded by the robotic swarm is the density of black spots, which may vary from trial to trial in the range $d \in [0, 1]$. Robots can perceive the colour of the ground only locally, through a noisy infrared sensor placed under their chassis. Robots can emit flashing signals, which can be perceived by all other robots. By combining the locally acquired information through this kind of simple communication, the group should encode the global density through the frequency of the emitted signals: the higher the density, the higher the frequency of the collective flashing signal. In the following, we give the details of the experimental setup and of the evolutionary algorithm we used to synthesise the robot neural controllers.

2.1 The Robots and the Environment

The experimental arena is square (side $l = 2m$) and surrounded by walls. Circular black spots are painted on the ground in order to probabilistically obtain a desired global density. The spots are homogeneous in colour and size (radius $r = 2.5cm$), and are aligned to a square grid of 40×40 cells (see Fig. 1). The density d represents the probability that each cell of the grid is filled with a black spot. Therefore, when the density is 0, no spot is present and the arena ground is completely grey; when the density is 1, the arena is completely filled with black circular spots. In this way, we can control the black spot density with a single parameter, and we can create multiple instances for the same macroscopic value.

Ten robots (radius $3.75cm$) are randomly placed in the environment. Each robot is equipped with two wheels that provide a differential drive motion (maximum linear speed: $v_{max} = 8.2cm/s$). Robots can perceive walls and other obstacles by means of eight infrared sensors placed around the turret (see Fig. 2(a)). The infrared sensors can be exploited for obstacle avoidance. The ground colour is perceived through an infrared sensor placed under the chassis of the robot, in the front part (indicated by ‘G’ in Fig. 2(a)). In the absence of noise, the ground sensor returns 0 when is over a black spot, and 0.5 when is over the grey background. Additionally, we make this sensor very unreliable by adding 30% white noise to the absolute sensor reading. Finally, each robot r can emit a flashing signal $S_r(t)$ switching on for a time-step ($\Delta t = 0.1s$) the LEDs

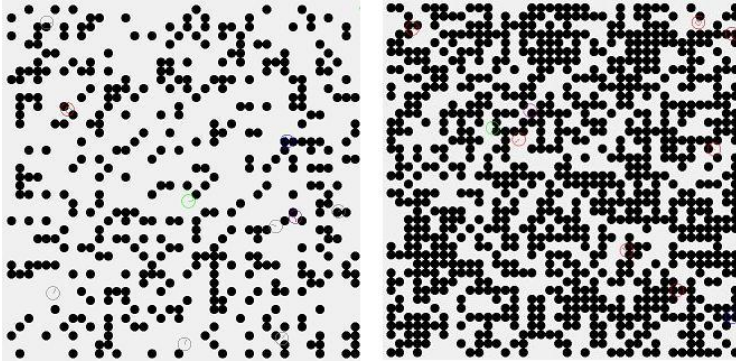


Fig. 1. Two snapshots of the simulated arena are shown. The black disks spots are painted on a grey floor. The spots are positioned on a grid of 40×40 cells. The density (i.e. the probability to find a spot in a cell) varies in the range $[0, 1]$ (left: $d = 0.26$, right: $d = 0.66$).

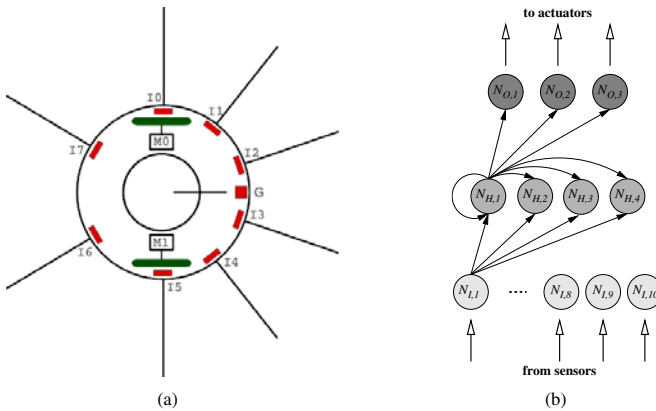


Fig. 2. The robot and the neural controller. (a) a schema of the simulated robot. Eight proximity sensors (IR_{0-7}) are positioned at 3.25cm from the ground pointing horizontally. The sensors detect obstacles at a maximum distance of $\approx 5\text{cm}$. The floor colour is perceived through an infrared sensor (G) positioned on the robots’ front and pointing the floor. (b) the agents’ controller is a CTRNN with 10 sensory neurons, 4 hidden and three motor units.

placed around its turret. This signal can be perceived by all the other robots present in the environment in a binary way: $s(t) = 1$ if there is at least one robot r emitting a signal, otherwise $s(t) = 0$. A robot can perceive the flashing signals through the omni-directional camera, including the signals emitted by the robot itself.

2.2 The Controller and the Evolutionary Algorithm

Each robot is controlled by a continuous time recurrent neural network (CTRNN) [15]. The neural network has a multi-layer topology, as shown in Fig. 2(b): neurons $N_{I,1}$ to $N_{I,10}$ take input from the robot’s sensory apparatus, neurons $N_{O,1}$ to $N_{O,3}$ control

the robot's actuators, and neurons $N_{H,1}$ to $N_{H,4}$ form a fully recurrent continuous time hidden layer. The input neurons are simple relay units, while the output neurons are governed by the following equations:

$$o_j = \sigma(O_j + \beta_j), \quad (1)$$

$$O_j = \sum_{i=1}^4 W_{ij}^O \sigma(H_i + \beta_i), \quad (2)$$

$$\sigma(z) = (1 + e^{-z})^{-1}, \quad (3)$$

where, using terms derived from an analogy with real neurons, O_j and H_i are the cell potentials of respectively output neuron j and hidden neuron i , β_j and β_i are bias terms, W_{ij}^O is the strength of the synaptic connection from hidden neuron i to output neuron j , and o_j and $h_i = \sigma(H_i + \beta_i)$ are the firing rates. The hidden units are governed by the following equation:

$$\tau_j \dot{H}_j = -H_j + \sum_{i=1}^4 W_{ij}^H \sigma(H_i + \beta_i) + \sum_{i=1}^{10} W_{ij}^I I_i, \quad (4)$$

where τ_j is the decay constant, W_{ij}^H is the strength of the synaptic connection from hidden neuron i to hidden neuron j , W_{ij}^I is the strength of the connection from input neuron i to hidden neuron j , and I_i is the intensity of the sensory perturbation on neuron i . The weights of the connection between neurons, the bias terms and the decay constants are genetically encoded parameters. Cell potentials are set to 0 each time a network is initialised or reset. State equations are integrated using the forward Euler method with an integration step-size of 0.1 seconds.

Eight input neurons— $N_{I,1}$ to $N_{I,8}$ —are set from the infrared sensors. Input neuron $N_{I,9}$ is set from the ground sensor. Finally, input neuron $N_{I,10}$ is a binary input set by the perception of the flashing signal $s(t)$. The neurons $N_{O,1}$ and $N_{O,2}$ are used to set the speed of the robot's wheels. Neuron $N_{O,3}$ is used to switch on the LEDs. In order to emit a flashing signal that lasts a single time-step, the LEDs are switched on only when the neuron activation surpasses the threshold 0.5:

$$S_r(t) = 1 \iff o_3(t) \geq 0.5 \wedge o_3(t-1) < 0.5. \quad (5)$$

This means that in order to flash again, the activation o_3 of neuron $N_{O,3}$ must go below the threshold, and up again. The minimum period for oscillations is therefore 2 time-steps, that is, 0.2s.

The free parameters of the robot's neural controller are encoded in a binary genotype, using 8 bits for each real number. Evolution works on a population of 100 randomly generated genotypes. After evaluation of the fitness, the 20 best genotypes survive in the next generation (elitism), and reproduce by generating four copies of their genes with a 2% mutation probability of flipping each bit. The evolutionary process lasts 5000 generations. During evolution, genotype parameters are constrained to remain within the range $[0, 1]$. They are mapped to produce CTRNN parameters with the following ranges: connection weights $W_{ij} \in [-4, 4]$; biases $\beta \in [-4, 4]$; concerning decay constants, the genetically encoded parameters are firstly mapped onto the

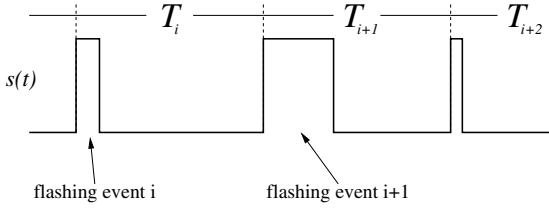


Fig. 3. Schematic representation of the collective flashing signal, through which the group of robots encodes the black spot density

range $[-1, 2]$ and then exponentially mapped onto $\tau \in [10^{-1}, 10^2]$. The lower bound of τ corresponds to the integration step size used to update the controller; the upper bound is arbitrarily chosen.

2.3 The Fitness Function

A genotype is translated into $N = 10$ identical neural controllers which are downloaded onto N identical robots (i.e., the group is homogeneous). Each group of robots is tested for 20 trials, which last either 1000 or 2000 time-steps (one time-step corresponds to 0.1s). The density is varied systematically, making the group experience 20 different values, equally distributed in $[0, 1]$. The robots' neural controllers are not reset from trial to trial, therefore the order in which trials are presented is relevant. At each fitness evaluation, we randomly shuffle the sequence of environments experienced by the same group, in order to remove regularities that could be exploited by spurious behaviours.

In order to evaluate the fitness of a genotype, we measure how well the corresponding robotic group encodes the black spot density d . To do so, we demand that robots as a group display a periodic flashing activity with a frequency that correlates with the black spot density. The group flashing activity is measured on the global signal $s(t)$ that results from the coupled activity of each robot. When robots flash in subsequent time-steps, their signals are perceived as a single *flashing event* (i.e., a sequence of consecutive flashes is perceived as a square signal, see Fig. 3). We measure the period T_i as the time between the start of two subsequent events. In this way, we obtain a series of inter-flash periods that we use to compute the fitness. First of all, we compute through an exponential moving average the average period \hat{T} and the average difference between two consecutive periods ΔT :

$$\hat{T} = \alpha \hat{T} + (1 - \alpha)T_i, \quad (6)$$

$$\Delta T = \alpha \Delta T + (1 - \alpha)|T_i - T_{i-1}|, \quad (7)$$

where $\alpha = 0.9$ is the time constant of the moving average. At the end of the trial θ , \hat{T} should encode the density. We measure the encoded density by linearly scaling the average period:

$$d_{enc} = \frac{T_M - \hat{T}}{T_M - T_m}, \quad (8)$$

where $T_M = 5s$ and $T_m = 1s$ are respectively the maximum and minimum periods, arbitrarily chosen. Finally, the two fitness components are computed: F_d^θ rewards the group for suitably encoding the black spot density:

$$F_d^\theta = \Phi(1.0 - |d - d_{enc}|), \quad (9)$$

where $\Phi(x)$ is a piecewise linear function that simply constrains the fitness value in the interval $[0, 1]$. This component therefore rewards the group for minimising the difference between the black spot density and the group encoded density. However, it does not assure that the system converges towards a periodic signalling. For this purpose, a second fitness component is computed, that minimises the difference between consecutive periods:

$$F_\Delta^\theta = \Phi(1.0 - \frac{\Delta T}{\Delta T_M}), \quad (10)$$

where $\Delta T_M = 2s$ is the maximum difference allowed. By minimising the difference among consecutive periods, the system is rewarded to produce periodic signals. Finally, the fitness of a genotype is the product of the two fitness components, averaged over multiple trials:

$$F = \sum_{\theta=1}^{20} F_d^\theta \cdot F_\Delta^\theta. \quad (11)$$

A trial is stopped and the fitness is zero when no flashing event is detected within the last 10s, therefore promoting a sustained flashing activity during the whole trial. Similarly, a trial is stopped if any robot collides with another robot or with a wall, and the fitness is zero for that trial. This indirect selective pressure allows to evolve obstacle avoidance.

3 Results

We have performed 20 evolutionary runs for 5000 generations. For each evolutionary run, we selected a single genotype to be further analysed. To this aim, we evaluated the performance of the 20 best individuals of the last generation, measuring the fitness over 100 trials for each of the 20 density values (2000 trials in total), and we selected the genotype with the highest mean performance to represent the evolutionary run. Among the selected genotypes, 9 out of 20 resulted in a good collective behaviour while the remaining ones resulted in sub-optimal solutions, in which the group always converged to a fixed signalling frequency, therefore failing to suitably encode the black spot density. The performance of the best genotypes is presented in Table [1](#). Despite the variability in performance, the behaviours evolved in different evolutionary runs are qualitatively similar: robots mainly rotate on the spot, in some cases slightly moving away from the initial position. While rotating on the spot, the ground sensor positioned on the robot front gives a very local and noisy estimate of the ground colour. The ground information is integrated over time, and modulates an internal oscillator that allows to tune the frequency of a periodic signalling. However, this frequency is related just to the local density perceived by the robot, which may be significantly different from the global density: in fact, an individual robot rotating in one place can perceive only a limited

number of different ground patterns, which do not represent well the global density, above all for intermediate density values. Moreover, the 30% white noise of the ground sensor makes it difficult to have even a good and stable local perception. For these reasons, robots have to coordinate to better estimate the global density, and to do so, they can exploit the flashing signals.

Table 1. Performance of the genotypes that result in a good collective perception behaviour. Data are sorted in decreasing order and, for each column, the mean and standard deviation are shown. The columns represent the fitness F and the the two components F_d and F_Δ .

run	F	F_d	F_Δ
4	0.87 ± 0.06	0.92 ± 0.06	0.95 ± 0.02
19	0.85 ± 0.08	0.92 ± 0.08	0.93 ± 0.04
14	0.84 ± 0.07	0.89 ± 0.08	0.94 ± 0.03
9	0.83 ± 0.07	0.92 ± 0.06	0.91 ± 0.05
20	0.83 ± 0.08	0.92 ± 0.07	0.90 ± 0.05
10	0.82 ± 0.06	0.91 ± 0.07	0.90 ± 0.03
15	0.81 ± 0.07	0.88 ± 0.08	0.92 ± 0.03
3	0.80 ± 0.10	0.85 ± 0.11	0.94 ± 0.03
13	0.80 ± 0.07	0.89 ± 0.08	0.89 ± 0.04

By analysing the communication strategies evolved in the different evolutionary runs, we found that they can be grouped into two classes. In some cases, the flashing signals are *excitatory*, that is, signal reception anticipates or provokes the signal production. This is the case for the behaviour evolved in runs 4, 9, 10 and 15. In the other cases—namely, runs 3, 13, 14, 19 and 20—flashing signals are *inhibitory*, that is, signal reception prevents or delays the signal production. In order to understand the mechanisms behind these two strategies, we analyse the best performing genotype of each class, namely the one obtained in run 4 for the excitatory strategy, and the one obtained in run 19 for the inhibitory one.

3.1 Analysis of the Excitatory Strategy

To better understand the properties of the evolved behaviour, we first perform a generalisation test that aims at revealing how well the system behaves with varying densities. For this purpose, we have recorded the performance of the system over 50 different black spot densities uniformly distributed in the range $[0, 1]$ (200 trials per density value). The results are displayed in Fig. 4. Here, we plot the fitness F and the two components F_d and F_Δ for each density. Moreover, we also plot the encoded density d_{enc} . The figure reveals that the system shows a good behaviour for almost all densities. In average, performance F oscillates in the interval $[0.8, 1.0]$. Moreover, it is possible to observe that the component F_Δ is very high, especially for high densities. This indicates that the group is able to converge to a very regular flashing activity, while for smaller values the period is noisier. However, for small density values the component F_d is higher, revealing that the system performs better in this conditions. The actual abilities of the robotic group can be discussed looking at the encoded density d_{enc} , which

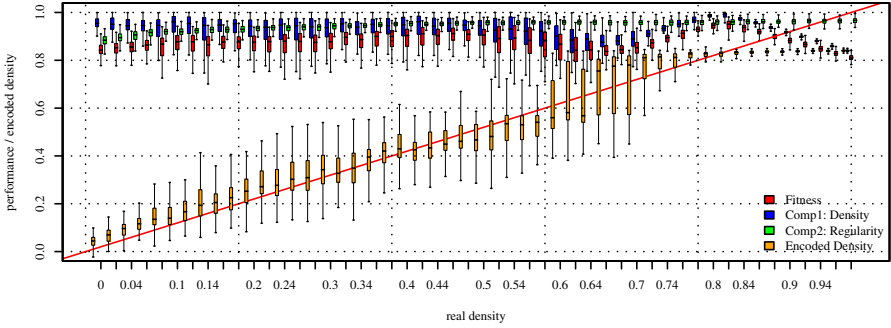


Fig. 4. Generalisation test for the excitatory strategy. Boxes represent the inter-quartile range of the data, while the horizontal lines inside the boxes mark the median values. The whiskers extend to the most extreme data points within 1.5 times the inter-quartile range from the box. The empty circles mark the outliers.

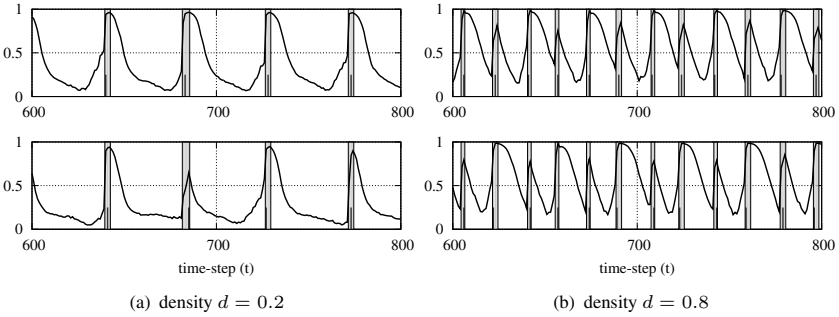


Fig. 5. Excitatory strategy: neural activation and signalling status for two of the ten robots. The bold line indicates the activation of the neuron $N_{O,3}$, which controls the flashing signal. The vertical grey bands indicate the perceived signal $s(t)$. The small dark vertical lines within the grey band indicate the time-step in which the robot itself is signalling.

is plotted against the ideal case $y = x$. We note that for densities up to $d = 0.6$, the encoded density nicely follows the real one. For larger values, however, a sort of phase transition occurs, in which the robots present a fast signalling behaviour that encodes a density around 0.84.

How can the robots produce such behaviour? We answer this question by analysing the behavioural and communication strategy. In this case, robots rotate on the spot without changing position. By observing an isolated robot, we noticed that the flashing activity is regulated by the locally perceived density: the higher the density, the higher the flashing frequency. However, the individual robot always underestimates the real density, in average. Therefore a collective mechanism must be in place. As mentioned above, in the excitatory strategy the reception of a signal provokes the emission of a signal. The dynamics of the oscillation for a low density $d = 0.2$ can be observed looking at Fig. 5(a). In correspondence of a perceived signal, the activation o_3 of the neuron controlling the signal output increases until it goes beyond the 0.5 threshold, making the

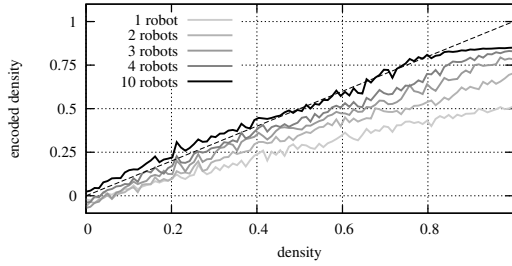


Fig. 6. Encoded density for varying group size. Each line represents the average of 100 trials, performed for 100 density values in $[0, 1]$. Data for 1, 2, 3, 4 and 10 robots are shown.

robot itself signal. Upon the sustained perception of a signal, the activation o_3 remains high, therefore delaying the following flash. For instance, in the first signalling event in Fig. 5(a) the top robot flashes the earliest, and the persistence of the signal afterwards delays the following flash. Instead, if more than one flash is required for o_3 to overcome the signalling threshold, the following flash is anticipated: the bottom plot reveals that in correspondence of a very delayed flash—during the second signalling event—the activation o_3 is just over the threshold and goes immediately down afterwards, allowing the robot to anticipate the following flash. The sequence of perceived flashes functions both as a positive and negative feedback mechanism: robots compete in emitting the first flash, and consequently mutually accelerate their rhythm. This acceleration is however limited by the presence of multiple signals that slow the flashing frequency down. The same mechanism is in place for larger frequencies (see Fig. 5(b)). However, in this case the system converges into a different dynamic regime, in which robots differentiate in two groups that alternately signal. This is evident in the dynamics of the activation o_3 shown in Fig. 5(b): the asymmetric oscillations indicate that robots engage in a sort of turn-taking, achieving the maximum flashing frequency. This also justifies the phase transition we observed in Fig. 4: for high densities the probability of converging into this fast flashing regime is higher. In order to further test the hypothesis that robots compete to emit the first signal, we run a series of experiments varying the number of robots in the arena. The results plotted in Fig. 6 show that the average encoded density increases with the number of robots, thus suggesting that robots are able to collectively accelerate their flashing rhythm.

3.2 Analysis of the Inhibitory Strategy

In the case of the inhibitory strategy, we performed similar analyses. The results of the generalisation test are plotted in Fig. 7. It is possible to notice that the system has a very similar performance with respect to the excitatory case: the performance F_d is very high for each density, while F_Δ slightly increases for large d . Therefore, also in this case the group converges towards very regular and precise flashing activity, especially for high densities. Looking at d_{enc} , it is possible to notice that the system presents a phase transition similar to the one discussed for the excitatory strategy.

All these similarities, however, result from radically different mechanisms. As we already mentioned, in this case signals are inhibitory: when a robot perceives a flash,

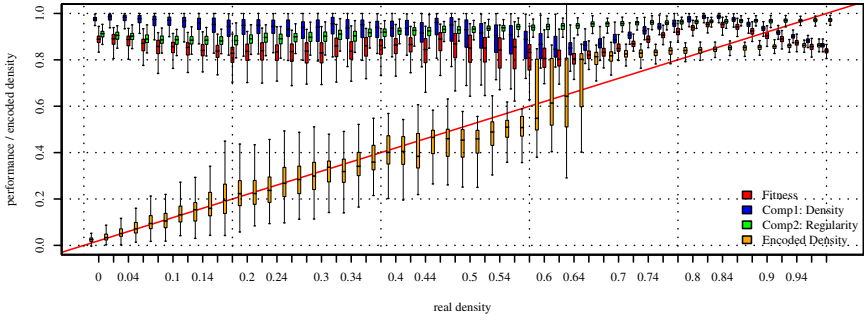


Fig. 7. Generalisation test for the inhibitory strategy

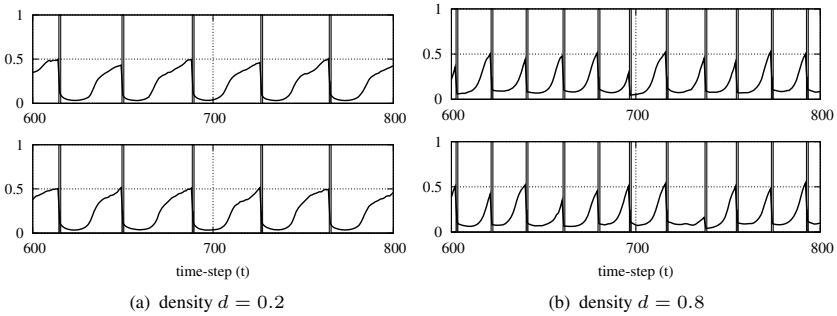


Fig. 8. Neural activation and signalling status for the inhibitory strategy

the neural activity o_3 that controls the flashing signal is reset, whatever its value is. This means that there is normally only one robot flashing at any time, that is, the one that reaches the signalling threshold the earliest. This behaviour is evident looking at Fig. 8(a), in which the dynamics of the neural activity of two different robots are plotted for a density $d = 0.2$: the bottom plot reveals that the corresponding robot flashes the earliest in the first four signalling events, preventing other robots to flash themselves. The situation is similar in the case of $d = 0.8$, shown in Fig. 8(b), in which we observe that robots compete in order to flash the earliest, similarly to what happens for the excitatory strategy. However, in this case the inhibitory signal does not allow a negative feedback mechanism. In fact, if a robot flashes with an individual frequency higher than the other robots (e.g., the robot locally perceives a higher density), it would impose its frequency to the group by inhibiting all other robots. If this is the case, the group systematically overestimates the black spot density due to those robots that locally perceive a high value. Therefore there must exist another mechanism that serves as negative feedback to control the frequency of the group. By looking at the behaviour of the robots, we notice that at the beginning of the trial robots slightly move from their initial position while rotating on the spot. This allows robots to explore the neighbourhood for a short time. In order to understand the role of these movements, we tested the robotic system fixing the motor outputs to constant values ($o_1 = 1$ and $o_2 = 0$), forcing the robots to turn on the spot without changing position. The results are shown in Fig. 9.

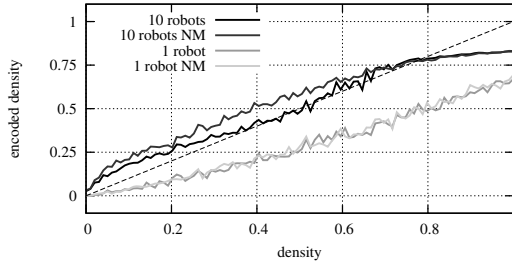


Fig. 9. Encoded density for varying group size and for blocked . Each line represents the average of 100 trials, performed for 100 density values in $[0, 1]$. Data for 1, 2, 3, 4 and 10 robots are shown.

for varying density and varying number of robots. From these tests, we infer that the slight motion of the robots is an adaptive mechanisms, given that the system without motion performs worse. As predicted, when robots cannot search the neighbourhood of their initial position, they slightly overestimate the density. We also observe that robots exploit the information coming from other robots. In fact, there is no difference between the performance of a single robot when it can and when it cannot move. This means that the motion alone does not allow a single robot to better estimate the global density. We therefore believe that the initial motion of the robots is performed when there are discrepancies between the locally perceived density and the global flashing activity. In other words, a robot moves in search of a local density that corresponds to the globally encoded density. This is brought forth only for a short time at the beginning of the trial. After this short time, the robot stops in place, whatever the local density is. With this mechanism, robots can average out the global density.

4 Discussions and Conclusions

In this paper, we have analysed how a group of robots can collectively encode a macroscopic variable that is not accessible to the single individuals in the group. By evolving the collective perception behaviour, we have found two possible strategies that use the communication channel in a opposite way: signals are either excitatory or inhibitory. In both cases, robots compete in flashing the earliest. In doing so, they share the information gathered locally, allowing to collectively encode an average value close to the actual density.

It is important to remark the fact that, besides the excitatory or inhibitory communicative interaction, a second mechanism is necessary to regulate the activities of the group. On the one hand, this mechanism has been found in the length of the signalling event, which limits robots in producing the first flash for many times consecutively. On the other hand, we observed that robots move from their initial position, therefore exploring the neighbourhood. In both cases, these regulatory mechanisms allow multiple robots to participate in the collective perception in order to have a better estimate of the macroscopic variable. It is therefore possible to identify a general strategy that supports the collective perception in our system: individually, robots encode the local density in a

flashing frequency and compete in producing the first flash, which is globally perceived and influences the whole group. At the same time, robots try to hand on the leader role and to listen to the other robots. The balancing of these two tendencies leads to the correct encoding of the global density.

The presence of two counteracting mechanisms that regulate the activity of the group is common to systems as diverse as brains and swarms. The positive feedback loop allows to amplify small perturbations and quickly spread information in a system, while the negative feedback loop controls the competition between different options and modulate the information spreading. The relevance of the negative feedback is recognised in neural systems—in which it is provided by specialised inhibitory inter-neurons and mediated by glycine and gamma-aminobutyric acid (GABA) transmitters [16]—and in super-organisms—in which it may result from specific stop signals issued by some individuals [17]. In our system, we have not provided a specific interaction modality different from the flashing signals. Despite this limitation, evolution could synthesise other mechanisms that resulted in regulatory processes.

In future work, we plan to continue the study of cognitive abilities displayed by collective systems. The experimental scenario we have presented here has been conceived to investigate the collective perception and decision making. We plan to study whether groups of robots can select the most dense environment among two or more possibilities presented sequentially or segregated in space. By comparing the results obtained in different artificial setups, we aim at discovering general principles about collective perception and decision making that could be generalised also to the biological reality [18].

Acknowledgements. The authors thank the Institute of Cognitive Sciences and Technology of the Italian National Research Council for having funded the research work presented in this paper. The authors also thank the members of LARAL group for the constructive comments during the early preparation of this research work.

References

1. Romo, R., Salinas, E.: Flutter discrimination: neural codes, perception, memory and decision making. *Nature Reviews Neuroscience* 4, 203–218 (2003)
2. Löffler, G.: Perception of contours and shapes: Low and intermediate stage mechanisms. *Vision Research* 48, 2106–2127 (2008)
3. Leopold, D.A., Logothetis, N.K.: Activity changes in early visual cortex reflect monkeys' percept during binocular rivalry. *Nature* 379, 549–553 (1996)
4. Rubin, N.: Binocular rivalry and perceptual multi-stability. *Trends in Neurosciences* 26, 289–291 (2003)
5. Grill-Spector, K.: The neural basis of object perception. *Current Opinion in Neurobiology* 13, 159–166 (2003)
6. Dehaene, S.: The neural basis of the weber-fechner law: a logarithmic mental number line. *Trends in Cognitive Sciences* 7, 145–147 (2003)
7. Hölldobler, B., Wilson, E.O.: *The Superorganism: The Beauty, Elegance, and Strangeness of Insect Societies*. W. W. Norton & Company, New York (2008)
8. Detrain, C., Deneubourg, J.L.: Self-organized structures in a superorganism: do ants “behave” like molecules? *Physics of Life Reviews* 3, 162–187 (2006)

9. Passino, K., Seeley, T., Visscher, P.: Swarm cognition in honey bees. *Behavioral Ecology and Sociobiology* 62, 401–414 (2008)
10. Marshall, J.A.R., Bogacz, R., Dornhaus, A., Planqué, R., Kovacs, T., Franks, N.R.: On optimal decision-making in brains and social insect colonies. *Journal of the Royal Society Interface* 6, 1065–1074 (2009)
11. Ratcliff, R., Smith, P.L.: A comparison of sequential sampling models for two-choice reaction time. *Psychological Review* 111, 333–367 (2004)
12. Trianni, V., Nolfi, S.: Self-organising sync in a robotic swarm. a dynamical system view. *IEEE Transactions on Evolutionary Computation, Special Issue on Swarm Intelligence* 13, 722–741 (2009)
13. Sperati, V., Trianni, V., Nolfi, S.: Self-organised path formation in a swarm of robots. *Swarm Intelligence* 5, 97–119 (2011)
14. Hauert, S., Zufferey, J.C., Floreano, D.: Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots* 26, 21–32 (2009)
15. Beer, R.D.: A dynamical systems perspective on agent-environment interaction. *Art. Intell.* 72, 173–215 (1995)
16. Jonas, P., Buzsaki, G.: Neural inhibition. *Scholarpedia* 2, 3286 (2007)
17. Nieh, J.C.: Negative feedback signal that is triggered by peril curbs honey bee recruitment. *Current Biology* 20, 310–315 (2010)
18. Trianni, V., Tuci, E.: Swarm cognition: an interdisciplinary approach to the study of self-organising biological collectives. *Swarm Intelligence* 5, 3–18 (2011)

Solving SONET Problems Using a Hybrid Scatter Search Algorithm

Anabela Moreira Bernardino¹, Eugénia Moreira Bernardino¹,
Juan Manuel Sánchez-Pérez², Juan Antonio Gómez-Pulido²,
and Miguel Angel Vega-Rodríguez²

¹ Computer Science and Communication Research Centre
School of Technology and Management, Polytechnic Institute of Leiria, 2411 Leiria, Portugal
{anabela.bernardino, eugenia.bernardino}@ipleiria.pt

² Department of Technologies of Computers and Communications, Polytechnic School
University of Extremadura, 10071 Cáceres, Spain
{sanperez, jangomez, mavega}@unex.es

Abstract. Nowadays the Synchronous Optical NETWORKing (SONET) standard is widely used in telecommunications. In this paper we consider three problems that arise in SONET networks: (1) Weighted Ring Edge-Loading Problem (WRELP), (2) SONET Ring Assignment Problem (SRAP) and, (3) Intraring Synchronous Optical Network Design Problem (IDP), known to be NP-hard. WRELP asks for a routing scheme such that the maximum load on the edges of a ring will be minimum. In SRAP the objective is to minimise the number of rings (i.e., DXCs). In IDP the objective is to minimise the number of ADMs. The last two problems are subject to a ring capacity constraint. We study these three problems without demand splitting and for solving them we propose a Hybrid Scatter Search (HSS) algorithm. Coupled with the Scatter Search algorithm, we use a Tabu Search algorithm to locate the global minimum. We show that HSS is able to achieve feasible solutions, improving the results obtained by previous approaches.

Keywords: Communication networks, Weighted ring edge-loading problem, SONET design problems, Scatter search algorithm, Bio-inspired algorithms.

1 Introduction

In this paper we consider the Weighted Ring Edge-Loading Problem (WRELP) which arises in engineering and planning of SONET rings [1]. Specifically, for a given set of non-splittable and unidirectional point-to-point demands, the purpose is to find the routing for each demand so that the maximum link segment load will be minimised [2], [3], [4], [5]. We verify that the main purpose of mainly previous works was to build feasible solutions for the loading problems in a reduced amount of time. Our purpose is different - we want to compare the performance of our algorithm with others in the achievement of the best-known solution.

In a SONET network, each customer (or node) is connected to one or more rings and the entire network is made up of a collection of such rings. The choice of

assigning a customer to a single ring or to multiple rings, and the way the rings are connected, determines different designing issues. In this paper we consider two different designing techniques determining different network topologies.

In the first topology we consider the SONET/SDH Ring Assignment Problem (SRAP). In this problem, each customer has to be assigned to exactly one SONET ring and a special ring called federal ring interconnects the other rings together through a special device, the Digital Cross Connect (DXC), which is the most costly network component. In this problem a capacity constraint on each ring is imposed. The capacity of each bidirectional ring of the network, including the federal ring has to accommodate the sum of bandwidth requests between all pairs of nodes connected by that ring. The problem is to find a feasible assignment of the customers minimising the total number of rings used (i.e. DXCs used) [5], [6], [7].

In the second topology, customers can be connected to more than one ring. If two customers want to communicate, they have to be connected to the same ring. In this case, the DXCs and the federal ring are not necessary, however the number of ADMs increase substantially in this topology. Since ADMs are the most expensive component in this network topology, it is important to obtain the smallest number of ADMs to reduce the cost of the network. This combinatorial optimisation problem is called Synchronous Optical Network Design Problem (IDP). Similarly to SRAP, the capacity of each ring is limited [6], [7], [8], [9].

Since the three problems studied in this paper are NP-hard combinatorial optimisation problems, we cannot guarantee to find the best solution in a reasonable amount of time. In practice, approximate methods are used to find a good solution to complex combinatorial optimisation problems where classical heuristics have failed to be efficient. The existing, successful methods in approximate optimisation fall into two classes: Local Search (LS) and population-based search. There are many LS and population-based optimisation algorithms. This paper presents an application of a population-based optimisation algorithm called the Scatter Search (SS) algorithm, combined with a LS technique called the Tabu Search (TS). The SS is an algorithm that has recently been found to be promising to solve combinatorial optimisation problems. The SS was first introduced in 1977 by Glover [10] and extensive contributions have been made by Laguna [11]. Embedded in the SS algorithm we use a TS algorithm, which is used to improve the solutions' quality. The TS algorithm is a mathematical optimisation method, which belongs to the class of LS techniques [12].

For the WRELP we compare the performance of Hybrid SS (HSS) algorithm with five algorithms: Probability Binary Particle Swarm Optimisation (PBPSO), Genetic Algorithm (GA), Hybrid Differential Evolution (HDE) algorithm, Hybrid ACO (HACO) algorithm and Discrete Differential Evolution (DDE). For SRAP and IDP we compare our results with: Diversification by Multiple Neighbourhoods (DMN and DMN2). For the SRAP we also compare our results with: standard GA, GA with Evolutionary Path-Relinking (GA-EvPR), Greedy Randomised Adaptive Search Procedure (GRASP), and GRASP with Path-Relinking (GRASP-PR). All the algorithms used for comparison were used in literature to solve the same problems.

The paper is structured as follows: in Section 2 we present the definition of the problems; in Section 3 we present the related literature; in Section 4 we describe the implemented HSS algorithm; in Section 5 we discuss the computational results obtained and in Section 6 we report about the conclusions.

2 Problems Definition

2.1 Weighted Ring Edge-Loading Problem

Let R_n be a n -node bidirectional SONET ring with nodes $\{n_1, n_2, \dots, n_n\}$ labelled clockwise. Each edge $\{e_k, e_{k+1}\}$ of R_n , $1 \leq k \leq n$, is taken as two arcs with opposite directions, in which the data streams can be transmitted in either direction: $a_k^+ = (e_k, e_{k+1})$ or $a_k^- = (e_{k+1}, e_k)$.

A communication request on R_n is an ordered pair (s, d) of distinct nodes, where s is the source and d is the destination. We assume that data can be transmitted clockwise or counter-clockwise on the ring, without splitting. We use $P^+(s, d)$ to denote the directed (s, d) path clockwise around R_n , and $P^-(s, d)$ the directed (s, d) path counter-clockwise around R_n . Often a request (s, d) is associated with an integer weight $w \geq 0$; we denote this weighted request by $(s, d; w)$. Let $Z = \{(s_1, d_1; w_1), (s_2, d_2; w_2), \dots, (s_m, d_m; w_m)\}$ be a set of integrally weighted requests on R_n . For each request (s_i, d_i) we need to design a directed path P_i of R_n from s_i to d_i . A collection $P = \{P_i; i=1, 2, \dots, m\}$ of such directed paths is called a routing for Z .

In this work, the solutions are represented using integer vectors (Table 1). If a position has a positive value, the demand flows in the clockwise direction, if it has the value 0, it flows in the other way. We assume that weights cannot be split, that is, for some integer $L_i \neq 0$, $1 \leq i \leq m$, the total amount of data is transmitted along $P^+(s_i, d_i)$; $L_i = 0$, the total amount of data is transmitted along $P^-(s_i, d_i)$. The vector $L = (L_1, L_2, \dots, L_m)$ determines a routing scheme for Z .

Table 1. Representation of the WRELP/SRAP/IDP solution

$Pair(s,d)/Edge(u,v)$	Demand					
1: (1, 2) \rightarrow	10		5: (2, 3) \rightarrow	10		
2: (1, 4) \rightarrow	20		6: (3, 6) \rightarrow	20		
3: (1, 7) \rightarrow	10		7: (3, 7) \rightarrow	10		
4: (1, 8) \rightarrow	30		... \rightarrow	...		
			14: (7, 8) \rightarrow	20		
$n=numberCustomers=8 \quad m=numberPairs=14 \quad B=150$						
Representation WRELP	$Pair_1$	$Pair_2$	$Pair_3$	$Pair_4$...	$Pair_{14}$
	10	0	20	0	...	0
Representation SRAP	$Customer_1$	$Customer_2$	$Customer_3$	$Customer_4$...	$Customer_8$
	1	2	2	1	...	1
Representation IDP	$Edge_1$	$Edge_2$	$Edge_3$	$Edge_4$...	$Edge_{14}$
	1	2	1	3	...	1

2.2 SONET Ring Assignment Problem

In literature there are different ways to represent a network. We consider the same network topologies described by Aringhieri and Dell'Amico [6] and we use the same model based on graph theory for the problems SRAP and IDP. In both topologies the objective is to minimise the total cost of the network and, at the same time is necessary to guarantee that the customer's demands, in term of bandwidth, are satisfied.

Considering a set of n customers and a symmetric traffic matrix $[d_{uv}]$, where $u, v = \{1, \dots, n\}$ and $u \neq v$, each entry of the matrix gives the amount of traffic between customer u and v . Note that $d_{uv} = d_{vu}$ and that $d_{uu} = 0$. Given an undirected graph $G = (V, E)$, the node set V contains one node for each customer and the edge set E contains one edge (u, v) for each pair of customers u, v such that $d_{uv} > 0$.

Problems SRAP and IDP correspond to two different partitioning of the above graph, subject to capacity constraints. In particular, SRAP involves a node partitioning, whereas IDP, involves an edge partitioning.

Formally, let V_1, V_2, \dots, V_k , be a partitioning of V in k subsets, the corresponding SRAP network is obtained by defining k local rings and a federal ring. All the customers of subset V_i are associated to the i -th local ring by means of an ADM and federal ring uses a DXC to connect each local ring. As a result the corresponding SRAP network uses n ADMs and k DXCs.

Solving SRAP corresponds to finding the partition V_1, \dots, V_k minimising k (number of rings), taking into consideration that the volume traffic on any ring is limited by a link capacity, called B . In other words, it is necessary to force the total traffic demands of all the customers connected to a ring to be lower or equal to the bandwidth (see Eq. 1) and also that the total traffic of the federal ring is not larger than the bandwidth B (see Eq. 2).

$$\sum_{u \in V_i} \sum_{v \in V, v \neq u} d_{uv} \leq B, \quad \forall i = 1, \dots, k \quad (1)$$

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^k \sum_{u \in V_i} \sum_{v \in V_j} d_{uv} \leq B \quad (2)$$

Likewise, for the SRAP, solutions are represented using integer vectors (Table 1). Each position of the solution corresponds to a customer and the value of that position corresponds to the ring to which it is connected. We assume that the weights of the traffic demands cannot be split and that each customer is assigned to one ring.

2.3 Intraring Synchronous Optical Network Design Problem

In IDP there is no need to assign each customer to a particular ring since customers can be connected to several rings. For this problem, the model is based on the edges of the graph, where a subset of the partition corresponds to a ring.

Given a partition of E into k subsets E_1, E_2, \dots, E_k , the IDP network can be obtained by defining k rings and connecting each customer of V (E_i) to the i -th ring by means of an ADM. In this case, the DXC are no longer needed and neither is the federal ring. Solving IDP corresponds to finding the partition E_1, E_2, \dots, E_k for each edge in order to minimise the number of ADMs [6], [7].

In this problem, like in SRAP, the traffic volume inside each ring is also limited by a link capacity B (see Eq. 3).

$$\sum_{(u,v) \in E_i} d_{uv} \leq B, \quad \forall i = 1, \dots, k \quad (3)$$

IDP solutions are also represented using integer vectors (Table 1), where each position of the solution corresponds to an edge and the value of that position corresponds to the partition to which it is associated. Also, in IDP, the weights of the traffic demands cannot be divided.

3 Related Literature

Cosares and Saniee [2], and Dell'Amico et al. [3] studied the non-split loading problem on SONET rings. Cosares and Saniee [2] proved that the formulation without demand splitting is NP-Hard. For the split problem, various approaches are summarised by Schrijver et al. [4] and their algorithms are compared in Myung and Kim [12] and Wang [13]. Recently, Kim et al. [14] presented an Ant Colony Optimisation algorithm using different strategies to solve the SONET ring loading. The main purpose of previous works was to produce near optimal solutions for WRELP in a reduced amount of time. Our purpose is different, - we want to compare the performance of our algorithm with others in the achievement of the best-known solution. Using the same principle, Bernardino et al. [15] proposed several bio-inspired algorithms to solve the non-split WRELP. The authors made a comparison between several heuristics to prove the efficiency of their algorithms. The same authors of this paper solved a similar WRELP problem - the Weighted Ring Arc Loading Problem (WRALP) that arises in Resilient Packet Ring systems [16]. In that paper the authors also proposed the HSS algorithm to solve the WRALP.

Goldschmidt et al. [9] studied the SRAP and shown that it is NP-hard. The authors proposed three polynomial-time heuristic algorithms to solve the problem, namely the edge-based, the cut-based and the node-based heuristic. Aringhieri et al. [17] solved SRAP with metaheuristic algorithms mainly based on Tabu Search. The authors introduced an objective function that depends on the current search status, and used a strategic oscillation obtained through the swap of two neighbourhoods. Macambira et al. [18] studied a similar SRAP problem - the k -SRAP problem. In k -SRAP the number of rings is fixed to k and it does not constrain the capacity of the federal ring to be B . The authors proposed an integer linear programming formulation to solve this problem. In [19] Macambira and Maculan reformulate SRAP as a set partitioning model with an additional knapsack constraint. To solve it, they implemented a branch-and-price/column generation algorithm. Bastos et al. in [20], [21], [22] solved SRAP using GRASP, GRASP with Path-Relinking and GA-EvPR.

In [8] and [23] Lee et al. and Laguna studied different formulations of the IDP and proposed several heuristics to solve these formulations. Goldschmidt et al. [9] considered the special case of IDP in which all the edges have the same weight. In [9] the authors show that this problem is NP-hard.

Aringhieri and Dell'Amico [6] studied the SRAP and the IDP as presented in this paper. They first described several objective functions that depend on the transition from one solution to a neighbouring one, then they applied several diversification and intensification techniques including two versions of Path Relinking (PR1 and PR2), eXploring Tabu Search (XTS) and Scatter Search (SS). They also proposed a Diversification method based on the use of Multiple Neighbourhoods (DMN). A set of extensive computational results is used to compare the behaviour of the proposed methods and objective functions. SRAP and IDP have been recently studied by

Pelleau et al. [7]. Their work extends the seminal ideas introduced by Aringhieri and Dell’Amico [6]. They proposed a new fitness function and a new local search algorithm (DMN2, a variant of the DMN proposed in [6]) to solve the problems and compared their method to previous ones. We use the fitness function proposed in this paper to evaluate SRAP and IDP solutions (see section 4.3).

4 Hybrid Scatter Search Algorithm

This metaheuristic technique derives from strategies proposed by Glover [10] to combine decision rules and constraints, and was successfully applied to a large set of problems [24]. The basic idea is to create a set of solutions (the reference set), that guarantees a certain level of quality and diversity. The iterative process consists in selecting a subset of the reference set, combining the corresponding solutions through a strategy, in order to create new solutions and to improve them through a LS optimisation technique. The process is repeated with the use of diversification techniques, until certain stopping criteria is met.

In SS algorithm an initial set of solutions (reference set) are build and then the elements of specific subsets of that set are systematically combined to produce new solutions, which hopefully will improve the best-known solution (see Glover et al. [24] for a comprehensive description of the algorithm). The basic algorithmic scheme is composed of five steps: (1) Generation and improvement of solutions; (2) Construction of the reference set; (3) Subset selection; (4) Combination; and (5) Reference set update. The standard SS algorithm stops when the reference set cannot be updated. However, the scheme can be enhanced by adding new steps in which the reference set is regenerated. Our algorithm uses a diversification mechanism after a pre-defined number of *nid* iterations without improving the best solution found so far. The reinitialisation can be very useful to refocus the search on a different search space region and to avoid the early convergence of the algorithm.

The main steps of the HSS algorithm applied to the problems are detailed below:

```

Initialise Parameters
Generate initial set of Solutions
Evaluate Solutions
Apply Improvement Method
Generate Reference Set
WHILE TerminationCriterion()
  Select subsets
  Apply Combination Method
  Apply Improvement Method
  Update Reference Set
  IF (no new solutions) THEN
    Regenerate Reference Set
  IF (nid iterations without improve best solution) THEN
    Apply Diversification Mechanism

```

The next subsections describe each step of the algorithm in detail.

4.1 Initialisation Parameters

The following parameters must be defined by the user: (1) *mi*– number of iterations; (2) *ti*– number of seconds; (3) *ni*– number of initial solutions; (4) *bl*– number of best

solutions in the reference set; (5) $b2$ – number of most different feasible solutions in the reference set and (6) nid – number of iterations without improvement (used for diversification).

4.2 Generation of Solutions

The initial WRELP solutions can be randomly created or in a deterministic form based in a Shortest-Path Algorithm (SPA). The SPA is a simple traffic demand assignment rule in which the demand will traverse the smallest number of segments. This rule has been used by other authors to create the initial set of solutions [15], [16].

For SRAP and IDP the solutions can also be created randomly or in a deterministic form. We create a simple heuristic that builds balanced solutions based on the number of customers for each ring. We consider an initial $k = k_{lb} + 1$ (see Eq. 4).

$$k_{lb} = \left\lceil \frac{\sum_{u=1}^{n-1} \sum_{v=u+1}^n d_{uv}}{B} \right\rceil \quad (4)$$

The main steps of the heuristic algorithm are detailed bellow:

```

FOR c=1 TO n DO
    customer[c] ← -1
END FOR
FOR r=1 TO k DO
    numberCustomers[r] ← 0
END FOR
c ← 0
WHILE c ≠ n DO
    cr ← Choose a random customer
    IF customer[cr] = -1 THEN
        customer[cr] ← c
        c ← c+1
    END IF
    FOR c=1 TO n DO
        rn ← choose a random ring
        FOR r=1 TO k DO
            IF numberCustomers[r] ≤ n/k THEN
                rn ← r
            END IF
        END FOR
        solution[customer[c]] ← rn
        numberCustomers[rn] ← numberCustomers[rn]+1
    END FOR
END WHILE

```

4.3 Evaluation of Solutions

To evaluate how good a potential solution is in relation to other potential solutions we use a fitness function. The fitness function returns a positive value (fitness value) that reflects how optimal the solution is. The fitness function used for WRELP is based on the fitness function used in [15], [16].

$$w_1, \dots, w_m \rightarrow \text{demands of the pairs } (s_1, t_1), \dots, (s_m, t_m) \quad (5)$$

$$V_1, \dots, V_m = 0 \rightarrow \text{Path}^-(s_i, t_i) \quad \text{or} \quad 1 \rightarrow \text{Path}^+(s_i, t_i)$$

$$\text{Load}(L, e_k) = \sum_{i: a_k^+ \in P^+(s_i, d_i)} w_i + \sum_{i: a_k^- \in P^-(s_i, d_i)} w_i \quad (6)$$

$$\forall k=1, \dots, n; \quad \forall i=1, \dots, m$$

$$\text{Fitness function: } \max\{\text{Load}(L, e_k)\} \quad (7)$$

For a given ring, between each node pair (s_i, t_i) there is a demand value ≥ 0 . Constraint sets (5) state that each positive demand value is routed in either clockwise (C) or counter-clockwise (CC) direction.

For an edge, the load is the sum of w_i for clockwise and counter-clockwise between nodes e_k and e_{k+1} (see Eq. 6). The purpose is to minimise the maximum load on the edges of a ring (see Eq. 7).

For SRAP and IDP, Aringhieri and Dell'Amico [6] introduced four objective functions. Let z_0 be the basic objective function counting the number of rings of a solution for SRAP, and the total number of ADMs for IDP, and let BN be the highest load of a ring in the current solution. $\alpha \geq 1$ and $\beta \geq 2$ are two fixed parameters, and $RingLoad(r)$ is the load of the ring r .

$$z_1 = z_0 + \max \{0, BN - B\} \quad (8)$$

$$z_2 = z_1 + \begin{cases} \alpha \cdot RingLoad(r) & \text{if the last move has created a new ring } r \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

$$z_3 = z_0 \cdot B + BN \quad (10)$$

$$z_4 = \begin{cases} z_{4a} = z_0 \cdot B + BN (= z_3) & \text{(a) : from feasible to feasible} \\ z_{4b} = (z_0 + 1)BN & \text{(b) : from feasible to unfeasible} \\ z_{4c} = z_0 B & \text{(c) : from unfeasible to feasible} \\ z_{4d} = \beta z_0 BN & \text{(d) : from unfeasible to unfeasible} \end{cases} \quad (11)$$

The first function z_1 (Eq. 8) minimises the basic function z_0 . If $BN > B$, it also penalises the unfeasible solutions. In addition to the penalty for the unfeasible solutions, z_2 (Eq. 9) penalises the moves that increase the number of rings. Function z_3 (Eq. 10) encourages solutions with small z_0 , while among all the solutions with the same value of z_0 , it prefers the ones in which the rings have the same loads. The last objective function z_4 (Eq. 10) is an adapting technique that modifies the evaluation according to the status of the search. It is a variable objective function having different expressions for different transitions from the current status to the next one.

In our work we used the fitness function z_5 (Eq. 12) introduced by Pelleau et al. [7]. The authors have implemented the five fitness functions. The z_5 function finds more good solutions than the other ones.

$$z_5 = z_0 + \sum_{p \in \text{partitions}} \text{violations}(p) \quad (12)$$

$$\text{violations}(p) = \begin{cases} \text{capacity}(p) - B & \text{if the load of } p \text{ exceed } B \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

The objective function z_5 minimises the basic function z_0 and penalises the unfeasible solutions. In this objective function every constraint violated (partition with total load higher than B) has a certain cost (the current load of a partition minus B). Summing all the violations of the current solution, the total violation is obtained. As z_0 is much smaller than the load of a ring, a feasible solution with 4 rings will be preferred to an

unfeasible solution with 3 rings. In z_5 partitions are all the rings (in the case of the SRAP the federal ring is also included).

4.4 Generation of Reference Set

The best $b1$ solutions in the initial set of solutions are selected to be in the reference set. The $b2$ feasible solutions in the initial set of solutions that are the most different when compared to the solutions already in the reference set, are also selected to be in the reference set. As a measure of the difference between two solutions, we compute the total number of different assignments between the two solutions.

4.5 Subset Selection

In literature, several methods can be applied to generate the subsets. In our implementation, the subsets are formed by combining two solutions from the reference set: $(1,2)$, $(1,3)$, $(1,4), \dots, (1,b1+b2)$, $(2,3), \dots, (b1+b2-1, b1+b2)$. We adopt Type-1 [24]. This method consists of $((b1+b2)^2 - (b1+b2))/2$ pair wise combinations of the solutions. All pairs of solutions in the reference set are selected for the combination procedure (see subsection 4.6).

4.6 Combination Method

This method combines the solutions in each subset to form new solutions.

For WRELP first a random node is chosen and then the pairs with that node are exchanged (see sec. "Combination Method" in [16]) between the two solutions. The combination method produces two combined solutions. For SRAP and IDP First a random ring/partition is chosen and then the customers/edges of that ring/partition are exchanged between the two solutions (see sec. "Combination Method" in [25]).

The combined solutions go through the improvement phase (see subsection 4.7).

4.7 Improvement Method

A TS algorithm is applied to each solution in the initial set of solutions in order to reduce its cost, if possible. After the combination, the TS algorithm is also applied to improve the quality of the combined solutions. The basic concept of TS was described by Glover [26]. TS allows the search to explore solutions that decrease the objective function value only in those cases where these solutions are not forbidden. This is usually obtained by keeping track of the action used to transform one solution into the next. When an action is performed it is considered tabu for the next T iterations, where T is the tabu status length. A solution is forbidden if it is obtained by applying a tabu action to the current solution.

In our implementation, the TS only exploits a part of the neighbourhood. The most common and simplest way to generate a neighbour for WRELP is to exchange the traffic direction of one request. In our implementation, some positions of the solution are selected and their directions are exchanged (partial search). This method was used for the WRALP and is summarised in [16]. The positions which directions are exchanged are classified as tabu attributes. A candidate can be chosen as a new current solution, if the positions which directions are exchanged are not the same as those in the tabu list.

In SRAP and IDP the simplest way to generate a neighbour is to swap two customers in the permutation. In this case the size of the neighbourhood is $n*(n-1)/2$, which would be large for large-scale problems. This would waste a lot of computing time. Other way to generate a neighbour is to assign one customer to other ring/partition. The size of the neighbourhood is $n*(k-1)$, which is also large for large-scale problems. In our implementation, we generate a neighbour by swapping two customers between two rings, $r1$ and $r2$ (randomly chosen). The algorithm searches for a better solution in the initial set of neighbours. If the best neighbour improves the actual solution, then the LS algorithm replaces the actual solution with the best neighbour. Otherwise, the algorithm creates another set of neighbours. In this case, one neighbour results on assigning one customer of $r1$ to $r2$, or $r2$ to $r1$. The neighbourhood size is $N(r1)*N(r2)$ or $N(r1)*N(r2) + N(r1)+N(r2)$.

The LS algorithm consists in the following steps:

```

r1 = random (number of customers)   r2 = random (number of customers)
NN = neighbours of ACTUAL-SOL (one neighbour results of interchange
      one customer of r1 or r2 with one customer of r2 or r1)
SOLUTION = FindBest (NN)
IF fitness(ACTUAL-SOL)<fitness(SOLUTION) THEN
  NN = neighbours of ACTUAL-SOL (one neighbour results of assign
      one customer of r1 to r2 or r2 to r1)
  SOLUTION = FindBest (NN)
  IF fitness(SOLUTION)<fitness(ACTUAL-SOL) THEN
    ACTUAL-SOL = SOLUTION
  END IF
ELSE
  ACTUAL-SOL = SOLUTION
END IF

```

The evaluation process is the most time-consuming step of the algorithm, which is usually the case in many real-life problems. For IDP and SRAP, our TS algorithm has some important improvements. After creating a neighbour, the algorithm does not perform a full examination to calculate the new fitness value; it only updates the fitness value based on the modifications made to create the neighbour. The running time is considerably reduced. The two rings which customers are exchanged are classified as tabu attributes. A candidate can be chosen as a new current solution if the rings which customers are exchanged are not the same as those in the tabu list. In our implementation, like in WRELP, we don't explore neighbours when the two rings chosen are in the tabu list. In aspiration, just the best neighbour not tabu with a fitness value lower than the best is selected.

The TS ends when a maximum number of iterations is reached. Based on preliminary observations, we consider 10 iterations. With a higher value of iterations, the algorithm slows down. We also observed that a high number of iterations does not produce significant better results.

Based on preliminary observations we consider $N/20$ elements for the tabu list [16].

The improved solutions are considered for inclusion in the reference set (see subsection 4.8).

4.8 Reference Set Update

The purpose is to maintain a good level of quality and diversity. We adopted the dynamic reference set update [24]. A new feasible solution immediately enters in the reference set, if its quality is better than the quality of the worst solution, or if its diversity is greater than the diversity of the less different solution. Solutions that are equal to others already in the reference set are not allowed to enter under any condition. If the reference set is not updated, then the algorithm restarts the reference set (see subsection 4.9).

4.9 Regeneration of Reference Set

The algorithm creates another set of solutions - P_s (with the same size of the initial set of solutions). The new solutions go through the improvement phase (see subsection 4.7). A new feasible solution immediately enters in the reference set, if its quality is better than the quality of the worst solution. The b_2 solutions with greater diversity are erased from the reference set and the b_2 feasible solutions in P_s that are the most different when compared to the solutions already in the reference set are selected to be in the reference set.

4.10 Diversification Mechanism

This mechanism restarts the best b_1 solutions in the reference set. The algorithm creates another set of solutions - P_d (with the same size of the initial set of solutions). The new solutions go through the improvement phase (see subsection 4.7). The best (b_1-1) solutions in P_d are selected to be in the reference set. For the following iteration, we kept the best solution.

4.11 Termination Criterion

The algorithm stops when a maximum number of iterations (mi) is reached or when a maximum number of seconds (ti) is reached.

5 Benchmark Instances

We evaluate the utility of the algorithm HSS used to solve WRELP using the same instances produced by Bernardino et al. [15], [16]. The studied examples arise by considering six different ring sizes – 5, 10, 15, 20, 25 or 30 nodes. The demand cases are: (1) complete set of demands between 5 and 100 with uniform distribution; (2) half of the demands in Case 1 set to zero; (3) 75% of the demands in Case 1 set to zero and; complete set of demand between 1 and 500 with uniform distribution. Last case was only used for the 30 nodes ring. It was studied 19 instances. For convenience, they are labelled C_{ij} , where $1 < i < 6$ represents the ring size, and $1 < j < 4$ represents the demand case. Other authors from literature used similar instances, but with smaller ring sizes or only with a low variance range for the demands [2], [3], [4].

For SRAP we used three sets of instances. The first one, class C1 has been introduced in [5]. They have generated 80 geometric instances, based on the fact that customers tend to communicate more with their close neighbours, and 80 random

instances. These subsets have both 40 low-demand instances, with a ring capacity $B = 155$ Mbs, and 40 high-demand instances, where $B = 622$ Mbs. The graphs generated have $|V| \in \{15, 25, 30, 50\}$. In the 160 instances, generated by Goldschmidt et al. [5], 42 have been proven to be unfeasible for SRAP by Aringhieri and Dell'Amico [6] using CPLEX 8.0. Class C2 was obtained by randomly modifying the 42 instances in C1 that are unfeasible for SRAP [6]. The authors have created 230 new and hard feasible instances. The characteristics of an instance in C1 or C2 can be deduced from its name. Instance names always start with two letters. The first letter is either G or R meaning that the instance belongs to the geometric or the random subdivision, respectively. The second letter is either L or H, depending if the ring capacity is 155 Mbs or 622 Mbs, respectively. The last set of instances, class C3, has been presented in [8]. They have generated 40 instances with a ring capacity $B = 48 * T1$ lines and the number of T1 lines required for the traffic between two customers has been chosen in the interval [1, 30]. The graphs considered have $|V| \in \{15, 20, 25\}$ and $|E| \in \{30, 35\}$. Most of the instances in this set are unfeasible for SRAP (only two are feasible).

For IDP we only studied the instances of the set C1 and C3. We want to compare our results with others from literature, and Pelleau et al. [7] provided their results for this two set of instances (they have specified the number of ADMs obtained for each instance). In [6] the authors also solved class C2, however they only presented the number of optimal solutions. To make an effective comparison it is necessary to know the exact number of ADMs obtained for each instance.

For more details on how these instances were generated and their properties, we guide the interested reader to the original papers where they were introduced.

6 Results

6.1 WRELP

The same authors of this paper have performed comparisons among all parameters (using all instances) of the HSS algorithm in order to establish the correct parameter setting for solving the WRALP [16]. The best results obtained with the HSS algorithm use ni between 40 and 100, $b1$ between 4 and 10, $b2$ between 4 and 10 and nid between $m/10$ and $m/2$. With these values, the algorithm reaches, in a reasonable amount of time, a reasonable number of best-known solutions. In general, the experiments have shown that the proposed parameter setting is very robust to small modifications. Based on preliminary studies [15] we verify that the behaviour of the WRELP is very similar to the WRALP, independent of the algorithm applied to solve it, so we used the same combination of parameters for this problem.

In this paper, we only compare our algorithm used to solve WRELP with: PBPSO, GA, HDE, HACO and DDE proposed by Bernardino et al. [15] to solve the same problem, because the authors: (1) use the same test instances; (2) adopt the same fitness function, and; (3) implement the algorithms using the same language (C++).

The six algorithms were executed using a processor Intel Quad Core Q9450. The initial solutions of the six algorithms were created using random solutions. For the instance C64 the SPA was used to create the initial populations.

Table 2. WRELP results – run times and number of iterations

Inst.	LS-PBPSO		GA		HDE		HACO		DDE		HSS	
	Time(s)	IT	Time(s)	IT	Time(s)	IT	Time(s)	IT	Time(s)	IT	Time(s)	IT
C11	<0.001	5	<0.001	2	<0.001	2	<0.001	4	<0.001	2	<0.001	2
C12	<0.001	1	<0.001	1	<0.001	2	<0.001	1	<0.001	1	<0.001	1
C13	<0.001	1	<0.001	1	<0.001	1	<0.001	1	<0.001	1	<0.001	1
C21	<0.001	20	<0.001	15	<0.001	10	<0.001	20	<0.001	15	<0.001	10
C22	<0.001	10	<0.001	10	<0.001	3	<0.001	10	<0.001	10	<0.001	3
C23	<0.001	5	<0.001	5	<0.001	3	<0.001	5	<0.001	5	<0.001	3
C31	0.1	30	0.1	30	0.1	15	0.1	30	0.1	15	0.1	15
C32	0.002	15	<0.001	20	<0.001	5	<0.001	15	<0.001	10	<0.001	5
C33	<0.001	5	<0.001	10	<0.001	5	<0.001	10	<0.001	5	<0.001	5
C41	0.15	100	0.15	100	0.1	30	0.15	100	0.1	50	0.1	30
C42	0.06	30	0.075	40	0.05	10	0.075	25	0.05	15	0.05	10
C43	<0.001	10	<0.001	10	<0.001	5	<0.001	10	<0.001	5	<0.001	5
C51	1.5	150	1.5	150	0.75	30	1.5	250	1.25	80	0.6	30
C52	0.15	40	0.15	60	0.1	15	0.15	50	0.15	25	0.1	15
C53	0.01	15	0.01	25	0.01	10	0.01	20	0.01	15	0.01	10
C61	2.5	200	2.25	150	1.75	40	2	200	2	80	1.5	30
C62	0.5	70	0.3	70	0.25	20	0.3	60	0.25	30	0.25	20
C63	0.075	15	0.075	30	0.075	10	0.075	20	0.075	15	0.05	10
C64	3	300	0.3	40	0.25	5	0.3	40	0.25	25	0.2	5

Table 2 presents the best WRELP results obtained with the six algorithms for the hardest instances. The first column represents the instance number (*Inst.*) and the remaining columns demonstrate the obtained results (*Time* – Run Times, *IT* – Iterations) by the six algorithms. The presented values have been computed based on 50 different executions for each test instance, using the best combination of parameters found and different seeds (see [15]). The six algorithms reach feasible solutions for all test instances and all the algorithms reach the best-known solutions (see [15]) before the run times and the number of iterations presented.

We must refer that our WRELP results are very similar to the WRALP results obtained in [16]. In fact, the formulation of the two problems is very similar. In [16] the authors proved that the standard deviations and average fitness for HSS are smaller for WRALP. It means that the HSS is more robust than the other algorithms. For the WRELP the algorithm produced similar results. In comparison, the HSS algorithm produces a higher number of best-known solutions using the same number of iterations. DDE algorithm obtains a good average fitness in a similar running time.

6.2 SRAP and IDP

Bernardino et al. studied a similar assignment problem in [25] – the Terminal Assignment Problem (TAP), a NP-Hard combinatorial optimisation problem. The main objective of this problem was to assign a collection of terminals to a collection of concentrators. In the paper, the authors have also proposed a HSS algorithm to assign terminals to concentrators. In a large network, some concentrators are used to increase the network efficiency. The terminals are connected to a concentrator and each concentrator is connected to the central computer. The purpose is to minimise the link cost to form a network by connecting a given set of terminals to a given set of concentrators. The number of concentrators and terminals and their locations are

known. Each concentrator is limited in the amount of traffic that it can accommodate. For that reason, each terminal must be assigned to one node of the set of concentrators, in such a way that no concentrator oversteps its capacity. We can establish an association of the TAP with the SRAP and IDP studied in this paper. Concentrators and terminals in TAP correspond to rings/partitions and customers in SRAP/IDP, respectively. Based in preliminary studies, we observed that the parameters values of the HSS in our problems have a similar influence.

Bernardino et al. [25] compared HSS with the TS, LSGA, HDE and HACO and all the five algorithms reach feasible solutions for all test instances. All the statistics obtained in [25] show that the performance of HSS algorithm is superior. The best results obtained with the HSS algorithm use ni between 30 and 100, $b1$ between 5 and 10, $b2$ between 5 and 10 and nid between $N/15$ and $N/2$. These parameters were experimentally considered good and robust for the instances tested.

In our paper, the parameters of the HSS algorithm were set to $ni=50$, $b1=8$, $b2=8$, and nid between $N/15$ and $N/2$. The initial population was created using the deterministic algorithm. Based on preliminary observations we verified that for each execution, using the same number of iterations or using a limited number of seconds the deterministic algorithm obtains a better average fitness in comparison with a random initial population.

Aringhieri and Dell'Amico [6] describe the results obtained for SRAP and IDP on the three benchmark sets studied in this paper, by the algorithms BTS, PR1, PR2, XTS, SS, and DMN. For each algorithm they consider four objective functions (z_1 , z_2 , z_3 and z_4), described in section 4.3. For the two problems, DMN with function z_4 gives the best overall results. Pelleau et al. [7] implemented all the algorithms described by Aringhieri and Dell'Amico, including a new one DMN2, and compare them using the new objective function z_5 . DMN and DMN2 were the heuristics that obtained the best results. Bastos et al. [20], [21], [22] solved SRAP using GA, GA-EvPR, GRASP and GRASP-PR, and obtained good results with them. In this paper we compare our IDP results with DMN and DMN2 (algorithms that obtained the best results in [7]), and our SRAP results with DMN, DMN2, GA, GA-EvPR, GRASP and GRASP-PR.

For each SRAP instance considered in our experiments, we fix a solution target value equal to the optimal solution. In [19], the authors implemented an exact algorithm that provided optimal solutions for the two classes C1 and C2 (in almost all instances $k_{lb}+1$). In class C3 there are only two feasible instances and for them we consider the lower bound $k_{lb}+1$. For IDP we consider the lower bound k_{lb} . We gave a time limit of 20 seconds to each run of our algorithm for solving SRAP, and one minute for IDP. However we observed that the average time to find the best solution is less than 2 seconds for SRAP and less than 10 seconds for IDP. Obviously, the algorithm terminates if the current best solution found is equal to the solution target. If the lower bound is not reached, we define as a high-quality solution, a solution for which the evaluation of the objective is equal to $k_{lb}+1$. To solve an IDP instance normally takes much more iteration to improve the value of the objective function than in SRAP. In SRAP the size of the solutions are smaller.

In Table 3, we compare the SRAP results obtained by HSS with the results e taken from [21], [22] and [7]. The first two columns show instance class and number of feasible solutions, the following columns give the percentage of optimal solutions found in literature by other algorithms. The presented HSS values have been

computed based on 50 different executions for each test instance, using the best combination of parameters found and different seeds (in almost all executions, the algorithm obtains the best solution).

Table 3. Results for SRAP

Class	FS	GRASP	GRASP-PR	GA	GA-EvPR	EB, CB, NB	DMN	DMN2	HSS
C1	118	100 *	100*	98,3	100	97,46	100	97,46	100
C2	230	98,69	99,57	85,7	98,26	-	98,26	-	99,57
C3	2	-	-	-	-	-	100	100	100

* The authors only consider 111 feasible instances

The HSS did not find the optimal solution for just one instance – new.GH_50_3.4. The optimal solution is 5 and our algorithm obtained the value 6.

For IDP we performed identical examinations. With HSS we have obtained the same number of ADMs for all the instances studied by Pelleau et al. [7] using DMN and DMN2. The authors indicate that for IDP problem, they have obtained better results for 15 instances. We have obtained the same values using smaller times for each execution. In their work they used a limit of 5 minutes. We have obtained our results with less than one minute. In 7 instances of the 200 instances studied for this problem, it was necessary $k_{ib}+1$ rings to obtain the smallest number of ADMs.

The results show that GRASP-PR and HSS present better results in terms of solution quality for SRAP and DMN, and HSS obtain better results also in terms of solution quality for IDP. We are not comparing computational times, because the experiments were performed on different equipment but we believe that our algorithm is very fast.

7 Conclusions

In this paper we present a HSS algorithm to solve the WRALP, SRAP and IDP. The HSS algorithm is an optimisation technique, able to perform simultaneous local and global search. Extensive computational experiments were done with benchmark instances from literature. The performance of HSS algorithm is compared with several algorithms used to solve the same problems studied in this paper.

The computational results show that HSS had a stronger performance, improving in some cases, the results obtained by previous approaches. We cannot say that for IDP and SRAP the HSS is the faster algorithm, because we have implemented and executed it in other platform, however we can say that it is competitive with the best heuristics in literature in terms of solution quality. We didn't find in literature all the necessary results to fully compare our results. It will be interesting to do it in the future.

In literature, the application of the HSS algorithm for these problems is nonexistent. For that reason, this article shows its enforceability in the resolution of these problems.

The continuation of this work will be the search and implementation of new methods to speed up the optimisation process.

References

1. Goralski, W.J.: SONET. McGraw-Hill Professional (2002)
2. Cosares, S., Saniee, I.: An optimisation problem related to balancing loads on SONET rings. *Telecommunication Systems* 3(2), 165–181 (1994)
3. Dell’Amico, M., Labbé, M., Maffioli, F.: Exact solution of the SONET Ring Loading Problem. *Oper. Res. Lett.* 25(3), 119–129 (1999)
4. Schrijver, A., Seymour, P., Winkler, P.: The ring loading problem. *SIAM Journal of Discrete Mathematics* 11, 1–14 (1998)
5. Goldschmidt, O., Laugier, A., Olinick, E.V.: SONET/SDH Ring Assignment with Capacity Constraints *Discrete. Appl. Math.* 129, 99–128 (2003)
6. Aringhieri, R., Dell’Amico, M.: Comparing Metaheuristic Algorithms for Sonet Network Design Problems. *Journal of Heuristics* 11, 35–57 (2005)
7. Pelleau, M., Van Hentenryck, P., Truchet, C.: Sonet Network Design Problems. In: *EPTCS 5, LSCS 2009*, pp. 81–95 (2009)
8. Lee, Y., Sherali, H.D., Han, J., Kim, S.: A Branch-and-Cut Algorithm for Solving an Intraring Synchronous Optical Network Design Problem. *Networks* 35, 223–232 (2000)
9. Goldschmidt, O., Hochbaum, D.S., Levin, A., Olinick, E.V.: The Sonet Edge-Partition Problem. *Networks* 41, 3–23 (2003)
10. Glover, F.: Heuristics for integer programming using surrogate constraints. *Decision Sciences* 8, 156–166 (1977)
11. Laguna, M.: Scatter search. In: Pardalos, P.M., Resende, M.G.C. (eds.) *Handbook of Applied Optimisation*, pp. 183–193. Oxford University Press (2002)
12. Myung, Y.S., Kim, H.G.: On the ring loading problem with demand splitting. *Operations Research Letters* 32(2), 167–173 (2004)
13. Wang, B.F.: Linear time algorithms for the ring loading problem with demand splitting. *Journal of Algorithms* 54(1), 45–57 (2005)
14. Kim, S.-S., Kim, I.-H., Mani, V., Kim, H.J.: Ant Colony Optimisation for SONET Ring Loading Problem. *International Journal of Innovative Computing, Information and Control* 4(7), 1617–1626 (2008)
15. Bernardino, A.M., Bernardino, E.M., Sánchez-Pérez, J.M., Vega-Rodríguez, M.A., Gómez-Pulido, J.A.: Solving ring loading problems using Bio-inspired algorithms. *Journal of Network and Computer Applications* 34(2), 668–685 (2011)
16. Bernardino, A.M., Bernardino, E.M., Sánchez-Pérez, J.M., Vega-Rodríguez, M.A., Gómez-Pulido, J.A.: Solving the ring arc-loading problem using a Hybrid Scatter Search Algorithm. In: *International Conference on Evolutionary Computation* (2010)
17. Aringhieri, R., Dell’Amico, M., Grasselli, L.: Solution of the sonet ring assignment problem with capacity constraints. Technical Report 12, DISMI. University of Modena and Reggio Emilia (2001)
18. Macambira, E.M., Meneses, C.N., Pardalos, P.M., Resende, M.G.C.: A novel integer programming formulation for the K-SONET ring assignment problem. *AT&T Labs Research Technical Report TD-6HLLNR* (2005)
19. Macambira, E.M., Maculan, N., Souza, C.C.: A column generation approach for SONET ring assignment. *Networks* 47(3), 157–171 (2006)
20. Bastos, L.O., Ochi, L.S., Macambira, E.M.: A relative neighbourhood GRASP for the SONET ring assignment problem. In: *Proceedings of the International Network Optimization Conference*, pp. 833–838 (2005)

21. Bastos, L.O., Ochi, L.S., Macambira, E.M.: GRASP with Path-Relinking for the SONET Ring Assignment Problem. In: Proc. Fifth International Conference on Hybrid Intelligent Systems (HIS 2005), pp. 239–244 (2005)
22. Bastos, L.O., Ochi, L.S.: A genetic algorithm with evolutionary path-relinking for the Sonet Ring Assignment Problem. In: International Conference on Engineering Optimization - EngOpt 2008, RJ. Proc. of the EngOpt 2008 - Sponsoring Societies: Mathematical Programming Society (MPS), ISSMO, EUROPT, ABCM. RJ : EngOpt, v. 1 (2008)
23. Laguna, M.: Clustering for the design of sonet rings in interoffice telecommunications. *Management Science* 40(11), 1533–1541 (1994)
24. Glover, F., Laguna, M., Marti, R.: Scatter Search and Path Relinking: Advances and Applications. In: *Handbook of Metaheuristics*, vol. 57, pp. 1–35. Springer, Heidelberg (2003)
25. Bernardino, A.M., Bernardino, E.M., Sánchez-Pérez, J.M., Vega-Rodríguez, M.A., Gómez-Pulido, J.A.: A Hybrid Scatter Search Algorithm to Assign Terminals to Concentrators. In: *IEEE Congress on Evolutionary Computation (CEC 2010)*, pp. 329–336. IEEE Computer Society, IEEE press, Los Alamitos (2010)
26. Glover, F., Laguna, M.: *Tabu Search*. Kluwer Academic Publishers (1997)

Investigating a Measure of the Recombinational Distance Traversed by the Genetic Algorithm

Robert Collier and Mark Wineberg

Department of Computing and Information Science
University of Guelph, Guelph, Ontario, Canada
{collierr,mwineber}@uoguelph.ca

Abstract. Measures of distance are essential to the development of many applications, but the need for these measures to be representative is often ignored - measures that truly represent the manner in which solution space is traversed are often disregarded in favour of simpler measures. With the genetic algorithm employing both unary and binary operators, it is difficult to quantify the distance between chromosomes with an approach that is truly representative of the distances traversed by the evolutionary mechanism. It is, however, possible to redefine the function of recombination to facilitate a more representative measure. The recursive approach presented here entails the redefinition of recombination as a set of unary operators determined by the current population. These operators replicate the behaviour of the original operator precisely and can be used to calculate the recombinational distance between chromosomes with a time complexity that is improved logarithmically over a simplistic approach.

Keywords: Genetic algorithms, Distance measurement, Complexity analysis.

1 Introduction

Whether conducting scientific studies on organisms that have been observed in the natural world, or developing simulations with which to analyze forms of artificial life, every scientist investigating the underlying mechanisms that govern the processes of evolution recognizes the need for scientific taxonomy and, ultimately, the importance of being able to quantify any distinguishing differences observed between organisms. However, where researchers of the natural world are largely restricted to collecting observations about the phenotypes of living organisms (often employing structures such as pedigree charts to trace evolutionary processes), for those researchers investigating the population simulations employed by the genetic algorithm it is possible to compute an accurate measurement of the distance between simulated chromosomes in terms of the actual genetic operators that are in use by the algorithm. Not only are these same measurements of distance essential for calculating population diversity, but any attempt to visualize the movement of a population through a search space of possible structures requires accurate and representative measures of interchromosomal distance.

As there are numerous applications for representative measures of interchromosomal distance (Stadler, 2002; Jones, 1995a; Wineberg and Oppacher, 2003), it is the objective of this paper to introduce and thoroughly explore an approach to the measurement of these distances with respect to the function of the recombination operator. Furthermore, as the incurrence of computational expense is often used in the justification of excessively simplistic methodologies, this paper places a strong emphasis on the complexity of the proposed approach. The details surrounding any one specific application of this measure largely exceed the scope of this paper and are only briefly addressed.

2 Genetic Operators

With the substitutional mutation process observed in natural world biology representing one of the simplest processes by which a new feature can be introduced into the phenotype of an organism, it is not surprising that most attempts to quantify the distance between chromosomes focus upon the distance that would be traversed by a point mutation operator. Mitchell (1998) offered the simple operational definition of the mutation operation of the genetic algorithm as the act of randomly changing the values of some alleles of a simulated chromosome. As it is technically possible, though highly unlikely, for every allele of a simulated chromosome to be mutated in a single generation of a typical genetic algorithm, it follows that it is then also possible for any chromosome to be entirely transformed into any other chromosome in a single generation. However, as the mutation operator is typically applied to each allele probabilistically and independently, the likelihood of one chromosome transforming into another decreases exponentially with the number of alleles that differ between the chromosomes in question. Consequently, the widely known Hamming distance metric, often used in quantifying the distance between two strings, is frequently employed by researchers of the genetic algorithm as a measurement of the distance between the chromosomes in the population simulation. Although the widespread use of the Hamming distance (Jones, 1995a) as a measure of the distance between simulated chromosomes is not inappropriate, it is important to acknowledge that Hamming distance alone is only representative of the developments facilitated by a mutation operator and, thus, should only be considered the sole source of variation in a population that employs asexual reproduction alone. With sexual reproduction becoming the predominant form of reproduction for the majority of the non-microscopic organisms observed in the natural world (Merrell, 1994), the genetic algorithm, seeking to emulate populations observed in the natural world as closely as possible, typically also employs a binary recombination operator that is often referred to as the crossover operator.

This recombination operator used by the genetic algorithm can be defined simply as an operator that exchanges data between the encoded chromosomes of two population members, in emulation of the biological process (Mitchell, 1998). Typically the operator randomly selects a set of alleles from one chromosome to be exchanged with the corresponding alleles of another. A uniform recombination operation will exchange each allele of a simulated chromosomes probabilistically and independently which, although similar to the manner in which the typical mutation

operator is applied, entails that the range of possible offspring that can be created through recombination is directly proportional to the genetic difference between the simulated chromosomes selected to act as parents.

Although k -point recombination operations, which randomly select a set number of substrings from a simulated chromosome for exchange, are also employed by genetic algorithm researchers with great frequency, since the set of possible offspring that can be created through the application of a uniform recombination operation contains all sets of possible offspring that can be created through the application of any number of fixed k -point recombination operations, for the sake of generalizability all subsequent references to recombination refer to uniform recombination.

3 Distance Measurement

The significance of distance functions to the genetic algorithm is most apparent when considering a formal definition of the fitness landscape (Stadler, 2002) that the genetic algorithm traverses in search of an optimum. Stadler defined the three-part composition of a fitness landscape to include an evaluation function to be optimized, the set of possible candidate solutions, that are represented by the genetic algorithm as simulated chromosomes, and a conceptualization of distance or neighbourhood that induces a topology on the solution set to create a solution space. Furthermore, knowing the distance between two chromosomes that must be traversed by the operators of the genetic algorithm is a reasonable indicator of the smallest number of generations it will take before the transformation of one chromosome to another is possible. Although the application of this information to optimization is apparent, by computing the distance between all possible pairs of chromosomes in the population, it is possible to get an impression of the actual diversity of the population as well.

For a function (whose domain is a pair of simulated chromosomes and whose range is a real value) to actually be considered a distance metric (as opposed to a distance measure), there are four conditions that must be satisfied. Firstly, the function must never report the distance between two elements of the solution set as a negative value, a condition known as non-negativity. The function must also comply with the identity of indiscernibles condition that states that the distance between two elements can and will only be considered zero if the two elements are identical. The third condition that must be observed, symmetry, states that the distance to be traversed from element x to element y must be the same as the distance that would be traversed from element y to element x . Finally, the function must also comply with the triangle inequality, which states that the distance from element x to element y must always be less than or equal to the sum of the distance from x to z and the distance from z to y . Since it is often the case that the mechanism of an operator cannot be described using a function that satisfies each of the four metric conditions presented above, a more generalized measure can be created by relaxing one or more conditions. Pseudometrics, semimetrics, and quasimetrics each observe three of the four conditions, failing to observe the identity of indiscernibles, the triangle inequality, and the symmetry conditions, respectively.

3.1 Recombinational Distances

The binary arity of the recombination operator, in contrast with the unary arity of the typical mutation operator, poses the most significant barrier to the introduction of an accurate measure of distance between chromosomes that would be traversed by the recombination operator. Since recombination requires two operand chromosomes to produce a single offspring chromosome, the notion of exactly two chromosomes being separated by any finite number of recombinations is undefined without the composition of the population. Consequently, when considering traversal of the search space using only recombination, the population must be explicitly considered, as in Fig. 1.

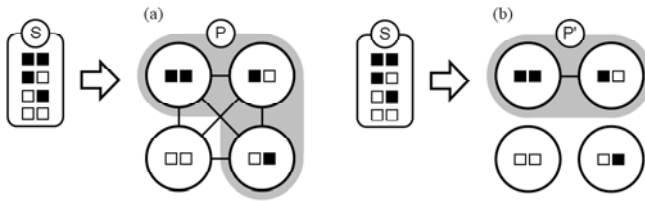


Fig. 1. Populations P and P', depicted as the shaded region of the solution space S of binary chromosomes of length 2. It is observed that, although the recombination operator can be applied to the chromosomes in the population P {■■, ■□, □■} depicted in Fig. 1. (a) to produce the chromosome □□ as an offspring (as is indicated by the edges between □□ and the shaded area P), if chromosome □■ is not present in the population, as is depicted in population P' in Fig. 1. (b), all edges incident on □□ disappear, demonstrating that the presence of an edge between chromosomes is dependent upon the entire population.

It was noted by both Jones (1995a, 1995b) and Culberson (1994) that considering each point in the search space to be a single chromosome does not permit researchers to explicitly connect them using a binary recombination operation. They proposed that points in the search space should represent possible chromosome pairings instead, between which connections would exist when one pair of chromosomes could be recombined to produce the other pair as offspring, as depicted in Fig. 2 on the following page.

Similarly, in Altenberg's (1997) development of an evaluation function for his fitness distance correlation counterexample, a measure termed "crossover distance" was defined as the number of single point recombination operations that must be applied to transform one pair of complementary chromosomes into another complementary pair. However, as recombination operations applied to complementary chromosomes can produce offspring of any genotypic configuration, this definition would not need to provide consideration for pairs separated by infinite distances. Although this was sufficient for the construction of Altenberg's function, it was also explicitly acknowledged that the recombination of complementary chromosomes is a rare occurrence during the operation of an actual instance of the genetic algorithm.

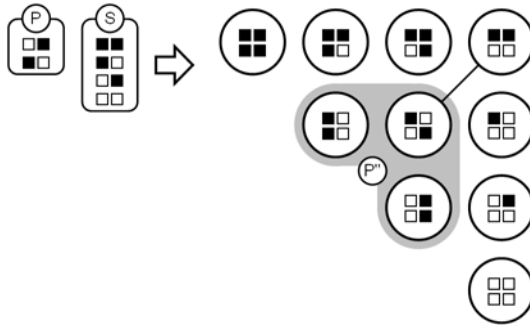


Fig. 2. It is possible to depict recombination operations in a simple graph if vertices represent pairs, rather than individual chromosomes. Although the space S of binary chromosomes remains unchanged from Figure 1, there are three unique pairings of the two members of population P , shaded and denoted P'' in the figure. The edge that connects pair $(\blacksquare\square, \square\blacksquare)$ with $(\blacksquare\blacksquare, \square\square)$ indicates that recombination between one pair could produce the other pair as offspring.

A contrasting alternative proposed by Gitchoff and Wagner (1996) employs a hypergraph topology wherein chromosomes are connected by as many hyperedges as there are offspring that could be the result of recombining hyperconnected chromosomes, as depicted in Fig. 3, below.

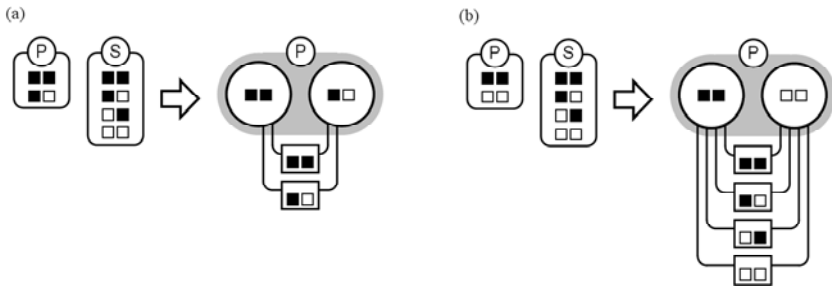


Fig. 3. Under the other paradigm proposed, possible recombination operations can be depicted in a hypergraph if vertices depict individual chromosomes connected by as many hyperedges as possible offspring, as demonstrated by the complementary pair $(\blacksquare\blacksquare, \square\square)$ in (b) having twice as many hyperedges as pair $(\blacksquare\blacksquare, \blacksquare\square)$ in (a)

4 Recombination Arity

Although either of the aforementioned techniques successfully captures some notion of the distance between possible chromosomes, a third alternative might suggest that the set of all possible binary recombination operations in a given population could instead be expressed using a set of unary operations. As a clarifying example, for a population of three simulated chromosomes, the set of possible binary operations $\text{recombine}(A, B)$, $\text{recombine}(A, C)$, and $\text{recombine}(B, C)$, could be equivalently expressed using the three unary operators recombineWithA , recombineWithB , and

recombineWithC. Under this paradigm, the distance between two simulated chromosomes with respect to traversal by the recombination operator would be the smallest number of unary recombination operations available within the current population. It is important to recognize, however, that the symmetry property normally associated with true measures of distances cannot be upheld when each binary recombination operation between two chromosomes is treated as a unary operation. Consider, as a clarifying example, sample chromosomes $A = \blacksquare\blacksquare\blacksquare\blacksquare$, $B = \square\square\square\square$, and $C = \blacksquare\blacksquare\square\square$ with the binary recombination operation redefined as two distinct unary operations. Although it is true that the operation $\text{recombineWithA}(B)$ is capable of producing an offspring chromosome C under uniform recombination, it does not follow that $\text{recombineWithA}(C)$ could produce B as an offspring. Since the distance (measured in terms of unary recombination operation recombineWithA) between B to C is finite while the distance from C to B is infinite, the recombination distance measure would, in fact, be more accurately defined as a quasimetric.

Although it is known that the search space of possible simulated chromosomes can only be depicted as a simple graph in two dimensions (with one vertex for each possible chromosome) if the undirected edges are representative of a unary operator such as mutation (Stadler, 2002), with the replacement of the binary recombination operator with a set of unary recombination operators, a graphical representation becomes possible. However, since it has been demonstrated that the unary recombination operator is not symmetric, a directed graph representation would be more accurate.

4.1 Unary Recombination Definition

In order to define a unary recombination operator it is first necessary to establish a definition of the space of possible chromosomes in terms of a single fixed chromosome, here denoted α , as was done with the recombineWithA operator example of the previous section. With the recombination operators of the genetic algorithm defined for chromosome operands of a fixed length λ , the set of possible chromosomes of the same length with which the fixed chromosome α could be recombined is referred to as the set β , of cardinality 2^λ . Within the set β there are exactly $C(\lambda, \delta)$ unique chromosomes at a Hamming distance of δ from the chromosome α , $\forall \delta$ where $0 \leq \delta \leq \lambda$. From the binomial theorem it is established that $\sum_{\delta=0}^{\lambda} C(\lambda, \delta) = 2^\lambda$ and, consequently, the subsets of β associated with each possible Hamming distance value of δ , for $0 \leq \delta \leq \lambda$, are mutually exclusive and exhaustive. Any chromosome β_i belonging to set β can be uniquely identified as the chromosome of length λ that has values complementary to those of α at the set of indices χ , where the cardinality of set χ can range from 0 (for chromosome β_1 , that is at a Hamming distance of 0 from α) to λ (for chromosome β_2^λ , that is complementary to chromosome α and, thus, at a Hamming distance of λ from α).

It is stressed that any binary string of length λ could be assigned to chromosome α provided that the set of chromosomes β is the set of all binary strings of length λ , ordered such that β_0 for $\chi = \{\}$ will be the binary string that is identical to α , having a Hamming distance of 0, β_1 for $\chi = \{1\}$ will be the binary string that is identical to α except at index 1 for which it will be complementary, having a Hamming distance of

1, etc. It is now possible to define a unary recombination operator such that the domain is a single chromosome and the range is a set of possible offspring chromosomes. The set of possible offspring chromosomes ϵ of a uniform recombination operation between parent chromosomes α and β_i is the set of chromosomes having values complementary to those of α at any set of indices that is a member of the power set $P(\chi)$. Equivalently, it could be stated that every element of the set of possible offspring chromosomes ϵ is contained within the highest order schema that contains both parent chromosomes α and β_i . This schema would only contain wildcard characters at indices where chromosomes α and β_i differ and, thus, the set of wildcard character indices would be equivalent to the set χ . For recombination between parent chromosomes α and β_i between which there is a Hamming distance value of δ , the cardinality of set χ will be δ , and thus the cardinality of the power set $P(\chi)$ will be 2^δ , as is evident in the example from Fig. 4.

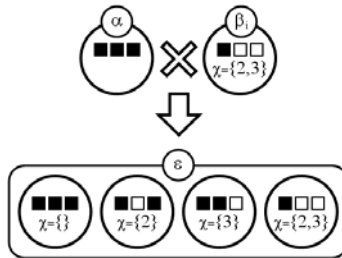


Fig. 4. For uniform recombination between the pair of parent chromosomes $\blacksquare\blacksquare\blacksquare$ and $\blacksquare\square\square$, where $\blacksquare\square\square$ is defined relative to $\blacksquare\blacksquare\blacksquare$ as having complementary index set $\chi = \{2, 3\}$, there exists exactly one possible offspring (defined relative to $\blacksquare\blacksquare\blacksquare$ with a complementary index set) for each a unique member of the power set of the complementary index set between α and β_i

It should be noted that since every chromosome β_i is described relative to chromosome α using a complementary index set χ , the actual configuration ($\blacksquare\blacksquare\blacksquare$) for the chromosome α need not have been explicitly noted. Had α been a different fixed chromosome ($\blacksquare\blacksquare\square$, for example), the complementary index set $\chi = \{2, 3\}$ would change the configuration of chromosome β_i (into $\blacksquare\square\blacksquare$ if α was configuration $\blacksquare\blacksquare\square$). The possible offspring would remain the configurations defined by complementary index sets $\{\}, \{2\}, \{3\}$, and $\{2, 3\}$. With every chromosome β_i described relative to α , it is sufficient to associate each set of possible offspring chromosomes, denoted ϵ , with the parent chromosome β_i which, when recombined with α , could produce those chromosomes as offspring.

With the newly established approach for redefining the space of possible chromosomes with respect to a single, fixed chromosome using complementary index sets, the set of unary recombination operators necessary to replace the binary recombination operator can be constructed. For every unique chromosome α in the population that could act as one operand of the binary recombination operator, there exists a unary operator (upon the chromosome space defined in terms of α) that takes a single operand chromosome and generates a set of possible offspring chromosomes

equal to the set of possible offspring for a binary recombination operation between the operand chromosome and the fixed chromosome α .

The associations present between chromosomes from set β and the set ϵ that represents the set of possible offspring of a recombination operation between a member of β and the chromosome α can be stored as an adjacency matrix that would define a directed graph structure representative of the recombination operations possible. Although similar to the matrix employed by Vose (1990) to encode mixing information (the probability that a pair of chromosomes, through both unary mutation and binary recombination, can produce a specific offspring), the adjacency matrix for the digraph representation of recombination would encode Boolean values for whether or not each chromosome could produce any other in the space solely through the act of recombining with a member of the population. Furthermore, as it was Vose's intention to employ the mixing probabilities in tandem with the selection probabilities (which cannot be computed without the evaluation function and a corresponding decrease in generality), for the present task of determining whether or not a given chromosome can be created by recombining elements of the current population, the proposed matrix of Boolean values would incur a lesser computational expense.

4.2 Digraph Representation

Since the recombination operations discussed herein probabilistically determine whether or not each allele of a chromosome will be exchanged independently, the adjacency matrix used to define the directed graph representation for recombination between chromosomes of length λ can be constructed recursively from adjacency matrices for chromosomes of length $\lambda-1$. Under the temporary assumption that chromosome α is the binary string of length λ comprised entirely of zero bits, there exists a $2^\lambda \times 2^\lambda$ matrix of Boolean values where entry ϕ_{ij} indicates whether or not recombination between α and the i^{th} member of the chromosome space can yield the j^{th} member of the chromosome space as an offspring. The matrix that would function as the basis for a recursive construction would be used for a chromosome length of 1 and, thus, entry ϕ_{00} would indicate whether or not chromosome α (which is \blacksquare) and the zeroth member of the chromosome space (which is also \blacksquare) can be recombined to produce the zeroth member of the chromosome space (which is also \blacksquare) as an offspring. Entry ϕ_{01} , on the other hand, would indicate whether or not chromosome α (which is \blacksquare) and the zeroth member of the chromosome space (which is also \blacksquare) can be recombined to produce the first member of the chromosome space (which is \square) as an offspring. For single bit chromosome recombination, the entries ϕ_{00} , ϕ_{01} , ϕ_{10} , and ϕ_{11} would be assigned the Boolean values true, false, true, and true, respectively.

For the recursive step in the construction of an adjacency matrix of the digraph representation for a chromosome of length λ , assume that the adjacency matrix of the digraph representation for a chromosome of length $\lambda - 1$ is complete and accurate. For entry ϕ_{ij} of the adjacency matrix for a chromosome of length λ to have a value of true, it must be possible to recombine the i^{th} member of the chromosome space of length λ , denoted " $i_1 i_2 i_3 \dots i_\lambda$ ", with a chromosome of length λ of only zero bits, such as $\blacksquare\blacksquare\blacksquare\dots\blacksquare$, and produce the j^{th} member of the chromosome space of length λ , denoted

" $j_1 j_2 j_3 \dots j_\lambda$ " as an offspring. In the case where $i_1 = \square$ this recombination is possible if and only if " $i_2 i_3 \dots i_\lambda$ " and $\blacksquare \blacksquare \blacksquare \dots \blacksquare$ can be recombined to produce " $j_2 j_3 \dots j_\lambda$ ", since an i_1 of \square can be recombined with a \blacksquare from α to produce either possible value of j_1 . Consequently, the $2^{\lambda-1} \times 2^{\lambda-1}$ entries ϕ_{ij} of the adjacency matrix for length λ for i from $[2^{\lambda-1}+1 \dots 2^\lambda]$ and j from $[1 \dots 2^{\lambda-1}]$ and the $2^{\lambda-1} \times 2^{\lambda-1}$ entries ϕ_{ij} of the adjacency matrix for length λ for i from $[2^{\lambda-1}+1 \dots 2^\lambda]$ and j from $[2^{\lambda-1}+1 \dots 2^\lambda]$ will both be precise copies of the adjacency matrix associated with chromosomes of length $\lambda - 1$. In the alternative case, where $i_1 = "0"$, recombination between " $i_1 i_2 i_3 \dots i_\lambda$ " and $\blacksquare \blacksquare \blacksquare \dots \blacksquare$ can only produce " $j_1 j_2 j_3 \dots j_\lambda$ " as an offspring chromosome if and only if $j_1 = \blacksquare$ and " $i_2 i_3 \dots i_\lambda$ " and $\blacksquare \blacksquare \blacksquare \dots \blacksquare$ can be recombined to produce " $j_2 j_3 \dots j_\lambda$ " as an offspring. Consequently, the $2^{\lambda-1} \times 2^{\lambda-1}$ entries ϕ_{ij} of the adjacency matrix for length λ for i from $[1 \dots 2^{\lambda-1}]$ and j from $[1 \dots 2^{\lambda-1}]$ will also be a copy of the adjacency matrix associated with length $\lambda - 1$ and the $2^{\lambda-1} \times 2^{\lambda-1}$ entries ϕ_{ij} of the adjacency matrix for length λ for i from $[1 \dots 2^{\lambda-1}]$ and j from $[2^{\lambda-1}+1 \dots 2^\lambda]$ will have a value of false.

For demonstrative purposes, consider the construction of the 4×4 adjacency matrix for the digraph representation of recombination applied to chromosomes of length 2. Under the continued assumption that chromosome α is comprised entirely of zero bits (in this case, chromosome $\blacksquare \blacksquare$), recombination with the 1st chromosome, $\blacksquare \blacksquare$, can produce only the chromosome $\blacksquare \blacksquare$ as an offspring. Thus, the first row of the adjacency matrix will be [true false false false]. Recombination between α and the second chromosome, $\blacksquare \square$, can produce either the $\blacksquare \blacksquare$ or the $\blacksquare \square$ chromosome as an offspring and, thus, the second row of the adjacency matrix will be [true true false false]. Similarly, the third and fourth rows of this adjacency matrix will be [true false true false] and [true true true true], respectively. The adjacency matrices constructed for the digraph representations of recombination operations that are applied to binary chromosomes of length 1 and 2 are depicted below in Fig. 5 (a) and (b), respectively.

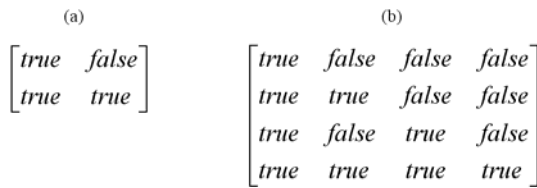


Fig. 5. The adjacency matrices used to define the digraph representation of recombination between chromosomes of length 1, (a), and length 2, (b). The recursive construction approach for these adjacency matrices is evidenced by the top left, bottom left, and bottom right quadrants of the matrix in (b) being identical to the matrix in (a).

As would be expected from the structural induction proof of the preceding paragraph, if the adjacency matrix associated with recombination on binary chromosomes of length 2 is bisected vertically and horizontally into exactly 4, 2×2 adjacency matrices, the top-left, bottom-left and bottom-right submatrices are copies of the basis matrix, and the top right submatrix is a 2×2 matrix comprised entirely of zeros.

It also follows that if the adjacency matrix for the digraph representation of recombination applied to chromosomes of length 3 is bisected vertically and horizontally into exactly 4, 4×4 adjacency matrices, the top-left, bottom-left, and bottom-right submatrices are each a copy of the adjacency matrix for the digraph representation of recombination applied to chromosomes of length 2, and the top right is a 4×4 matrix comprised entirely of zeros. Figure 6 clearly depicts the presence of the digraph representation associated with recombination for chromosomes of length 2 within the digraph representation associated with recombination for chromosomes of length 3.

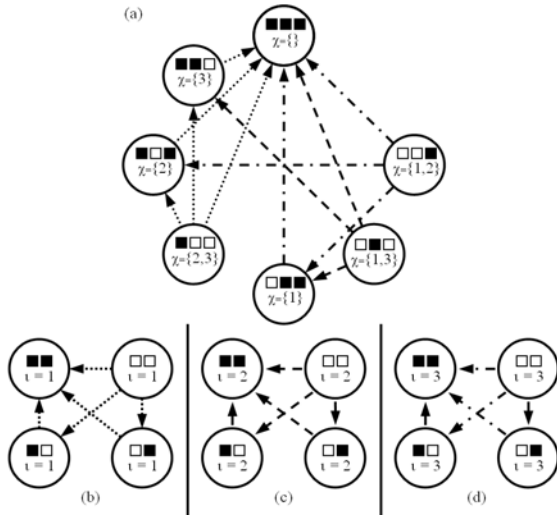


Fig. 6. For β_i separated from α by Hamming distance $\delta < \lambda$, the chromosomes must share at least one allele, making any recombination between these configurations equivalent to a recombination applied to configurations of length $\lambda-1$. In (a) above, since $\lambda = 3$, configuration β_i and α must share the symbol at index $\tau = 1, 2$ or 3 . If the index $\tau = 1$, then the digraph representation (b) of recombination for length 2 can be consulted, and the edges mapped to the nodes in (a) by inserting the symbol shared by β_i and α at index τ . For example, if β_i and α share the symbol at $\tau = 1$, the arc from $\square\square$ to $\blacksquare\blacksquare$ in (b) corresponds to the arc from $\blacksquare\square\square$ to $\blacksquare\blacksquare\blacksquare$. If the symbol at $\tau = 2$ is shared, the arc from $\square\square$ to $\blacksquare\blacksquare$ in (c) corresponds to the arc $\square\blacksquare\square$ to $\blacksquare\blacksquare\blacksquare$. Thus, every arc in a digraph representation for λ , except for those that would originate in the node complementary to α , can be determined from the digraph representation for $\lambda-1$.

5 Possible Offspring

It can be concluded, from the proof and discussion contained in the previous section, that if the first parent chromosome α of a recombination operation is a binary string of zero digits, there is a trivially simple recursive algorithm that will determine whether

the chromosome ϵ_i can be produced as an offspring of a recombination operation between the first parent chromosome α and the second parent chromosome β_i . This algorithm, in order to determine whether the i^{th} member of the chromosome space can produce the j^{th} member of the chromosome space as an offspring through recombination with a chromosome comprised entirely of zero bits, entails determining whether the entry ϕ_{ij} of the adjacency matrix lies in the top right quadrant of the adjacency matrix. If so, it can be concluded that a recombination operation between the i^{th} member of the chromosome space and the zero bit chromosome cannot produce the j^{th} member of the chromosome space as an offspring. If, however, the entry ϕ_{ij} lies in any other quadrant, the same algorithm is recursively applied to the 2^{nd} through the λ^{th} bits of chromosomes i and j until the chromosome length is 1.

5.1 Fixed Parent General Case

As an alternative to the development of a similar proof for every other possible value of the simulated first parent chromosome α , it would suffice to demonstrate that there exists a reversible transformation that, when applied to both the parent and offspring chromosomes, would convert one of the parent chromosomes into the binary string comprised entirely of zeros. Under this transformation, denoted τ , the Boolean value describing whether or not recombination between a pair of chromosomes β_i and β_j can yield chromosome ϵ_k as an offspring would be equivalent to the Boolean value describing whether or not a recombination operation applied to a chromosome α that is comprised entirely of zero bits and chromosome $\tau(\beta_j)$ can yield the chromosome $\tau(\epsilon_i)$ as an offspring.

Vose (1990) noted such a transformation in the second lemma of his technical report on the formalization of the genetic algorithm to be the application of the bitwise exclusive disjunction operator. This section will demonstrate that the use of this operator allows a single digraph representation of a recombination operation with a chromosome comprised entirely of zero bits to serve as a sufficient representation for any recombination operator.

If the previously mentioned adjacency matrix has already been constructed, wherein the Boolean value of entry ϕ_{ij} indicates whether or not recombination between a chromosome α comprised entirely of zero bits can be recombined with the i^{th} member of the chromosome space to yield the j^{th} member of the chromosome space as an offspring, then the question of whether uniform recombination between the pair of simulated chromosomes β_i and β_j can yield chromosome ϵ_k as an offspring is equivalent to the question of whether recombination between a chromosome α comprised entirely of zero bits and $\tau(\beta_j)$ can yield chromosome $\tau(\epsilon_i)$ as an offspring. This Boolean value, in turn, can be read directly from the adjacency matrix.

If transformation τ is the application of a bitwise exclusive disjunction operation (represented with the symbol \oplus) between the operand and the k^{th} member of the chromosome space, then $\tau("i_1 i_2 i_3 \dots i_\lambda")$ would be equivalent to " $k_1 \oplus i_1 k_2 \oplus i_2 k_3 \oplus i_3 \dots k_\lambda \oplus i_\lambda$ ". Since exclusive disjunction results in a value of false if and only if the two operands are either both true or both false, then " $\tau(k)1 \tau(k)2 \dots \tau(k)\lambda$ " would be equivalent to " $k_1 \oplus k_1 k_2 \oplus k_2 \dots k_\lambda \oplus k_\lambda$ ", also equivalent to $\blacksquare \blacksquare \blacksquare \dots \blacksquare$.

To solve for the Boolean value of whether recombination between the k^{th} and i^{th} member of the chromosome space, denoted " $k_1 k_2 k_3 \dots k_\lambda$ " and " $i_1 i_2 i_3 \dots i_\lambda$ " respectively, can produce the j^{th} member, denoted " $j_1 j_2 j_3 \dots j_\lambda$ ", as an offspring, the application of a bitwise exclusive disjunction operations with " $i_1 i_2 i_3 \dots i_\lambda$ " will transform the k^{th} , i^{th} , and j^{th} members of the chromosome space into configurations $\blacksquare \blacksquare \blacksquare \dots \blacksquare$, " $\tau(i)_1 \tau(i)_2 \tau(i)_3 \dots \tau(i)_\lambda$ ", and " $\tau(j)_1 \tau(j)_2 \tau(j)_3 \dots \tau(j)_\lambda$ ", respectively. It then suffices to prove that the Boolean value describing whether uniform recombination between configurations $\blacksquare \blacksquare \blacksquare \dots \blacksquare$ and " $\tau(i)_1 \tau(i)_2 \tau(i)_3 \dots \tau(i)_\lambda$ " can produce configuration " $\tau(j)_1 \tau(j)_2 \tau(j)_3 \dots \tau(j)_\lambda$ " as an offspring is equivalent to the Boolean value describing whether uniform recombination between the k^{th} and i^{th} member of the chromosome space can produce the j^{th} member of the chromosome space. For this to be true it must be shown that, for all values of x , $\tau(j)_y = 0 \vee \tau(i)_y$ will be true if and only if $j_x = k_x \vee i_x$ is also true. This particular fact can be most easily demonstrated by using a simple truth table to compare the values of $j_x = k_x \vee i_x$ and $\tau(j)_y = 0 \vee \tau(i)_y$ t, for all values of k_x , i_x , and j_x . This is included below as Table 1.

Table 1. The fact that the fourth column, $j_x = k_x \vee i_x$, and the eighth column, $\tau(j)_y = 0 \vee \tau(i)_y$, are equivalent demonstrates that recombination can produce offspring j from parent configurations i and k if and only if recombination between a chromosome comprised entirely of zeros and one equal to $i \oplus k$ can produce $j \oplus k$ as an offspring

k_x	i_x	j_x	$j_x = k_x \vee i_x$	$k_x \oplus k_x$ $\equiv \tau(k)_x$	$k_x \oplus i_x$ $\equiv \tau(i)_x$	$k_x \oplus j_x$ $\equiv \tau(j)_x$	$\tau(j)_y =$ $0 \vee \tau(i)_y$
0	0	0	true	0	0	0	true
0	0	1	false	0	0	1	false
0	1	0	true	0	1	0	true
0	1	1	true	0	1	1	true
1	0	0	true	0	1	1	true
1	0	1	true	0	1	0	true
1	1	0	false	0	0	1	false
1	1	1	true	0	0	0	true

5.2 Digraph Representation Properties

Since the set of possible offspring chromosomes that can be produced by the application of uniform recombination operations to chromosomes of length λ is equivalent to the set of possible chromosomes β with which chromosome α could be recombined to create offspring chromosomes, and since both sets are present in the digraph representation of recombination, the number of possible resultant offspring chromosomes is 2^λ . Furthermore, since the $C(\lambda, \delta)$ unique chromosomes at a Hamming distance of δ , where $0 \leq \delta \leq \lambda$, represent every possible chromosome with which chromosome α could be recombined, and since the cardinality of the set of possible offspring chromosomes that could be produced from a recombination operation applied to chromosomes between which there is a Hamming distance of δ is 2^δ , it stands to reason then that the number of arcs that are present in the offspring digraph is $\sum_{\delta=0}^{\lambda} C(\lambda, \delta) \cdot 2^\delta = (1+2)^\lambda = 2^\lambda$.

6 Complexity Impressions and Analyses

If the set of all possible chromosomes to be searched by the genetic algorithm is denoted R , it was explicitly observed by Jones (1995a, 1995b) and Culberson (1994) that binary recombination would then act on an element of R^2 to produce elements of R . This function could be accurately depicted using bipartite directed graph $G = (U, V, E)$ where, for every vertex of U representative of a pair of chromosomes, there exists an arc in E whose direct successor is a vertex in V representative of a chromosome that might be created by recombining the pair of chromosomes at the direct predecessor of the arc in U . While it is obvious that the cardinality of set V is the cardinality of the entire chromosome space S being searched, where $|S| = 2^\lambda$, depending upon whether or not the recombination operator is permitted to recombine a chromosome with itself, the cardinality of set U is, for a population containing exactly ρ unique chromosomes, either $(\rho+1)! / (2! \cdot (\rho-1)!)$ or $(\rho)! / (2! \cdot (\rho-2)!)$ respectively.

It might then be concluded that determining whether or not (from the set B of Boolean values) a specified chromosome (belonging to set V) can be produced by the application of a single recombination operation to a pair of chromosomes taken from the current population (belonging to set U), and, thus, evaluating the solution for the function $f:(U,V) \rightarrow B$, is actually equivalent to the task of searching the previously defined bipartite directed graph and must then have a time complexity of the order $O(\rho^2 2^\lambda)$.

The contrasting representation of binary recombination investigated by Gitchoff and Wagner (1996) employed a hypergraph wherein exactly one vertex exists for each possible chromosome, and a hyperedge between any two vertices would exist for each possible offspring that could be the result of a recombination operation between the hyperconnected vertices. Although this hypergraph would have only P vertices, the set of hyperedges that would connect a single pair of complementary vertices would have the cardinality of the entire chromosome space S . With binary recombination operations being possible between any two chromosomes in the population, this would be a complete graph of $n(n-1)/2$ edges, also suggesting a complexity of the order $O(\rho^2 2^\lambda)$.

6.1 Actual Complexity Analysis

With the proposed methodology, determining whether a given chromosome can be produced by a population through a single application of a binary recombination operator is equivalent to determining whether a given chromosome can be produced from any pair of chromosomes in the population, necessitating the $O(\rho^2)$ component of the complexity associated with examining all possible chromosome pairs. Although it remains true that recombination between a pair of complementary chromosomes could theoretically result in any chromosome in the search space S as an offspring, determining whether or not a matrix entry is located in the top right quadrant, at most λ times, has time complexity $O(\lambda)$.

Overall, the time complexity of the proposed recursive algorithm is the sum of the complexity of locating the appropriate matrix entries for all possible chromosome

pairings, $O(\rho^2\lambda)$, and the complexity of the application of the bitwise exclusive or operations necessary to redefine the chromosomes of the current population in terms of each possible fixed parent, also $O(\rho^2\lambda)$, for a total worst case time complexity of $O(\rho^2\lambda)$. Thus, the time complexity has been reduced from $O(\rho^{2^2\lambda})$ to $O(\rho^2\lambda)$, which constitutes a logarithmic speedup. Furthermore, for each of the λ determinations of whether the associated matrix entry lies in the top right quadrant of the adjacency matrix, the 25% likelihood that the algorithm can terminate early at every step of the recursion also indicates a very fast average case time complexity of the algorithm as well.

7 Discussion

It was previously noted that the notion of interchromosomal distances in the genetic algorithm is central to both the established adaptive landscape visualization technique and measures of population diversity. It was noted by Wineberg and Oppacher (2003) that every measure of population diversity in common usage is essentially an aggregating function of the Hamming distances between all possible pairs of chromosomes that are present in the population (or a slight variant thereof). Furthermore, when constructing a three-dimensional adaptive landscape visualization, the chromosome space must first be represented as a two-dimensional plane from which the landscape can be extruded.

Since the dimensionality of the chromosome space employed by a genetic algorithm is typically in excess of two, if researchers do not wish to limit their own usage of this visualization technique to instances where the evaluation function is of two dimensions or less the chromosome space dimensionality should be reduced by multidimensional scaling technique for which an accurate interchromosomal distance measure has been defined.

Although some researchers might consider the Hamming distance metric sufficient for calculating interchromosomal distances, it must be explicitly observed that the chromosome space is traversed by the mechanism of the genetic algorithm with both a mutation operator and a recombination operator, simultaneously. Since it has been previously demonstrated that recombination operations are more likely to assemble higher order building blocks than mutation operations (Spears, 1998), the set of approaches to interchromosomal distance measurement in the genetic algorithm would be remiss if a technique for measuring recombinational distance were not included.

8 Recent Advances

Since the analysis in this paper was initially presented at the International Conference on Evolutionary Computation in 2010, the authors have continued to investigate possible approaches to the measurement of recombinational distance, with the ultimate goal being the integration of both a mutational distance and a recombinational distance into a measure that is truly representative of the manner in which the space of candidate solution genotypes is traversed by a simple genetic algorithm. The approach introduced here was designed to generate a Boolean value

for whether or not a specified target chromosome could be produced by the application of recombination to those chromosomes that comprise the entire current population, but with the operational definition of the distance resolution as the cardinality of the total range of values that can be produced by the distance measure, the authors have been able to expand the approach to increase the distance resolution of the measure. To this effect, all possible subsets of the current population (sorted according to increasing cardinality) are also tested, in order to determine whether or not the target chromosome could be recombinationally produced as the offspring of an origin chromosome and a proper subset of the current population. The size of the smallest subset can then be reported as the recombinational distance from the origin chromosome to the target, except where a value of zero or $+\infty$ must be assigned, if the origin and target chromosomes are identical or if it is not possible to produce the target chromosome by recombining the origin chromosome with members of the population, respectively. Thus, the distance resolution of a recombinational distance measure can be increased from a value of two to a value of two plus the number of unique members of the population. Finally, the authors have been able to successfully integrate the above measure of recombinational distance with the common approach to measuring mutational distance, constructing a measure that is demonstrably more representative of the manner in which genotypic spatial distance is traversed by the genetic algorithm.

9 Conclusions

Although previous approaches to the depiction of the binary recombination operator would seem to suggest a time complexity $O(\rho^2 2^\lambda)$, this paper has demonstrated that a logarithmic speedup can be achieved. By first defining a set of unary recombination operators that are equivalent to the function of the binary recombination operator, followed by the application of a bitwise transformation on the operands, the time complexity associated with the process of determining whether a certain chromosome can be produced from a given population through a single recombination can be improved to $O(\rho^2 \lambda)$. The recursive approach presented in this paper affords researchers an opportunity to include consideration for the traversal of the chromosome space by both mutational and recombinational operations, which will ultimately result in more representative visualizations and calculations of population diversity.

Acknowledgements. The authors wish to acknowledge partial funding for this research by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Altenberg, L.: Fitness Distance Correlation Analysis: An Instructive Counterexample. In: Proceedings of the 7th International Conference on Genetic Algorithms, pp. 57–64 (1997)
2. Culberson, J.C.: Mutation-Crossover Isomorphisms and the Construction of Discriminating Functions. *Evolutionary Computation* 2, 279–311 (1995)

3. Dybowski, R., Collins, T.D., Weller, P.R.: Visualization of Binary String Convergence by Sammon Mapping. In: Proceedings of the 5th Annual Conference on Evolutionary Programming, pp. 377–383 (1996)
4. Gitchoff, P., Wagner, G.P.: Recombination Induced Hypergraphs. *Complexity* 2(1), 37–43 (1996)
5. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. (1989)
6. Hamming, R.: Error Detecting and Error Correcting Codes. *Bell System Technical Journal* 29(2), 147–160 (1950)
7. Jones, T.: *Evolutionary Algorithms, Fitness Landscapes, and Search*. Thesis Document. The University of New Mexico, Albuquerque (1995)
8. Jones, T.: *One Operator, One Landscape*. Working Paper. Santa Fe Institute (1995)
9. Merrell, D.J.: *The Adaptive Seascape: The Mechanism of Evolution*, p. 59 (1994)
10. Mitchell, M.: *An Introduction To Genetic Algorithms*. MIT Press, Cambridge (1996)
11. Sammon, J.W.: A Nonlinear Mapping for Data Structure Analysis. *IEEE Transactions on Computers* 18(5), 401–409 (1969)
12. Spears, W.M.: *The Role of Mutation and Recombination in Evolutionary Algorithms*. Thesis Document. George Mason University, Fairfax (1998)
13. Stadler, P.F.: Fitness Landscapes. *Biological Evolution and Statistical Physics*, 183–204 (2002)
14. Wijk, J.J.: The Value of Visualization. In: *IEEE Visualization Conference*, vol. 0, p. 11 (2005)
15. Vose, M.D.: Formalizing Genetic Algorithms. In: *Proceedings of Genetic Algorithms, Neural Nets, and Simulated Annealing Applied to Problems in Signal and Image Processing* (1990)
16. Wineberg, M., Oppacher, F.: The Underlying Similarity of Diversity Measures Used in Evolutionary Computation. In: *Proceedings of the 5th Genetic and Evolutionary Computation Conference*, pp. 1493–1504 (2003)
17. Wright, S.: The Roles of Mutation, Inbreeding, Crossbreeding and Selection in Evolution. In: *Proceedings of the 11th International Congress of Genetics*, vol. 8, pp. 209–222 (1932)

Enhancing the Adaptive Dissortative Mating Genetic Algorithm in Fast Non-stationary Fitness Functions

Carlos M. Fernandes^{1,2}, Juan Julián Merelo¹, and Agostinho C. Rosa²

¹ Department of Computers' Architecture, University of Granada, Granada, Spain

² LaSEEB-ISR-IST, Technical University of Lisbon, Lisbon, Portugal

{c.m.fernandes.photo, jjmerelo}@gmail.com

{acrosa}@laseeb.org

Abstract. The Adaptive Dissortative Mating Genetic Algorithm (ADMGA) is a variation of the standard GA in which a mating restriction based on the genotypic similarity of the individuals is introduced. The algorithm mimics a mating strategy often found in nature: dissimilar individuals mate more often than expected by chance and, as a result, genetic diversity throughout the run is maintained at a higher level. ADMGA has been previously applied to non-stationary fitness function, performing well when the changes hit the function at a medium and slow rate, while being less effective when the frequency is higher. Due to the premises under which the algorithm was tested, it has been argued that the replacement strategy that results from the implementation of the dissortative mating strategy may be harming the performance when solving high-frequency dynamic problems. This paper investigates alternative replacement strategies for ADMGA with the objective of improving its performance on this class of non-stationary problems. The strategies maintain the simplicity of the algorithm, i.e., the parameter set is not increased. The replacement schemes were tested in dynamic environments based on stationary functions with different frequency and severity, showing that it is possible to improve standard ADMGA's performance in fast dynamic problems by simple modifications of the replacement strategy.

Keywords: Genetic algorithms, Dissortative mating, Replacement strategies, Dynamic optimization problems.

1 Introduction

In the last two decades, Evolutionary Algorithms (EAs) [2] have been successfully applied to industrial problems, especially those with non-linearities and multiple objectives. However, real-world problems often have dynamic components that lead to (predictable or unpredictable) variations of the fitness function, i.e., the problem is defined by a time-varying fitness function. Such problems are called non-stationary (or dynamic) optimization problems.

A problem is said to be non-stationary when there is a change in the fitness function, problem instance or restrictions, thus making the optimum change as well. In each period of optimization, the fitness function is deterministic, but, when changes occur, solutions

already found may be no longer valid and the process must engage in a new search effort. EAs self-adaptive characteristics make them promising candidates to solve this type of problems. However, there other characteristics of these of these algorithms that demand a special care when tuning or designed them for tackling dynamic optimization problems. In nowadays, the study of efficient strategies for dynamic optimization is one the main research lines in the Evolutionary Computation field.

Recent studies on evolutionary dynamic optimization are mainly directed towards *diversity maintenance* techniques and *memory* schemes [5]. There are other possible approaches, like reacting to changes (by increasing mutation, for instance, as in [6]) when they occur, or even tackling the change with a new randomly generated population. However, the performance of such kind of approaches is strongly dependent on the intensity of the changes — they perform better when changes affect only a small percentage of the solution's variables — and, usually, require that the changes are easy to detect. Moreover, even if the change is easy to detect, it is not trivial to decide whether it is better to restart the population or to continue the search with the same population after a shift in the environment. Thus, it is sometimes better to have an algorithm that is capable of continuously adapting the solution to a changing environment — for instance, by using a *memory* [14, 18, 23] or a *multi-population* approach [4]. Memory may be very effective in some situations but their utility is believed to be restricted to a certain type of dynamics — in general, memory is particularly useful when the shape of the fitness landscape repeats from time to time. In addition, memory schemes require a considerable tuning effort and some parts of their design and implementation is not trivial.

Diversity maintenance techniques [12, 15, 23, 21, 13] do not require, in general, any knowledge about the problem and neither its dynamics nor its performance is reported to be highly dependent on a specific configuration of the problem. A possible approach for designing diversity maintenance EAs for dynamic optimization is using mating restrictions based on the genotypes. Dissortative mating, for instance, which refers to mating strategies in which dissimilar individuals mate more often than expected by chance, may be inserted into to an EA and slow down the diversity loss. There are several EAs in the literature with such type of mating strategies. One of them is the *Adaptive Dissortative Mating Genetic Algorithm* (ADMGA), proposed by Fernandes and Rosa in [10], and applied to dynamic optimization with promising results in [11] and [13]. However, it has been observed that its performance degrades when the frequency of changes increases. One of the possible explanations for this behavior resides in the replacement strategy and the premises under which it is tested: since changes are assumed to be hard to detect, the algorithm reevaluates every solution that remains in the population after one generation (Please note that this is the worst case scenario; in many applications the changes may be detected with less computational effort).

This problem arises because the original ADMGA's replacement procedure is a population-wide elitist strategy [20]: parents and children compete and *live* in the same population. It has been shown in [13] that if every old solution is reevaluated, then the average ratio between ADMGA's new individuals and function evaluations, in each generation, is approximately $1/2$. In addition, since the replacement strategy is elitist, it tends to reduce diversity. This may be slowing down ADMGA and the effect

is much more pronounced with high frequency problems because there are fewer evaluations than between each change.

This paper addresses this issue by proposing alternative replacement strategies that introduce diversity in the parents' subpopulation. Three different schemes are proposed: one in which the parents that remain in the population are first mutated and then reevaluated; another one that replaces the parents by mutated copies of the best individuals; finally, a third scheme, inspired by the *Random Immigrants Genetic Algorithm* (RIGA) [15], that replaces the parents that remain in the population by randomly generated solutions. The three strategies are tested in several dynamic problems constructed by applying the dynamic problem generator proposed in [22] to base-functions with binary variables. The results are compared to those attained by the standard ADMGA. Then, the best strategy is compared with a standard Generational Genetic Algorithm (GGA) and with the *Elitism-based Immigrants Genetic Algorithm* (EIGA) [23]. The results demonstrate that the best strategy is clearly capable of outperforming standard ADMGA on fast environments, without degrading its performance when the frequency is lower. Statistical tests are given.

The paper is structured as follows. The following section describes the most relevant dissortative mating strategies for EAs found in literature. Section 3 describes ADMGA and introduces the proposed replacement strategies. Section 4 describes the experimental setup and Section 5 presents and discusses the results. Finally, Section 6 concludes the paper and outlines future lines of research.

2 Background Review

By considering merely the quality of the solutions represented by the chromosomes when selecting individuals for mating purposes, the traditional EAs emulate what, in nature, is called *random mating* [19], i.e., mating chance is independent of genotypic or phenotypic distance between individuals. However, random mating is not the sole mechanism of sexual reproduction observed in nature. Outbreeding, assortative mating and dissortative mating [19] are all non-random strategies frequently found in natural species. Each one of these schemes has different effects on the genetic diversity of the population. Dissortative mating, for instance, increases the diversity of a population. Assortative mating, on the other hand, restricts mating between dissimilar individuals and leads to diversity loss.

Therefore, dissortative mating naturally came out in EAs' research field as an inspiration for dealing with the problem of premature convergence. A well-known EA with a dissortative mating strategy is the CHC [8]. CHC uses no mutation in the classical sense of the concept, but instead it increases the mutation probability when the best fitness does not change after a certain number of generations. A reproduction restriction assures that selected pairs of chromosomes reproduce unless their Hamming Distance is above a certain threshold, that is, the algorithm restricts crossover between similar individuals.

Another possible way of inserting assortative or dissortative mating into an EA is described in [9]. The *negative Assortative Mating GA* (nAMGA) selects one parent, by any method. Then, it selects a pool of p individuals (the size of the pool controls the intensity of the restriction) and computes the Hamming distance between those

chromosomes and the first parent. The individual less similar to the first parent is selected for recombination. Although nAMGA's results are interesting, the pool's size is critical to its performance and hard to tune.

Ochoa *et al.* [16] carried out an idea related with nAMGA in a dynamic optimization framework. Assortative and dissortative GAs are used to solve a dynamic knapsack problem. The results show that dissortative mating is more able to track solutions, while a standard GA often fails to track them. The assortative GA is the worst algorithm in the test set. The authors also discuss the optimal mutation probability for different strategies, concluding that the optimal value increases when the strategy goes from dissortative to assortative. In this line of work, there is also a study by Ochoa *et al.* [17] on the error threshold of replication in GAs with different mating strategies that aims at shedding some light into the relationship between mutation probabilities and mating strategies in EAs.

Besides the above-referred techniques, a large number of other GAs with non-random mating are found in the literature. Please refer to [13] for a detailed state-of-the-art review.

3 ADMGA and Replacement Strategies

There are many replacement strategies¹ for EAs but, in general, they may be classified into two categories: generational and elitist. Generational schemes replace the entire parents' population by the children; in elitist strategies, offspring must compete with their parents. ADMGA, due to its specific design, is a *population-wide* elitist strategy [20]. This means that some individuals may remain in the population for more than one generation. Since changes in non-stationary functions are not always easy to detect, the most reliable way to guarantee that a fitness value does not become outdated by a change in the environment is to reevaluate all the chromosomes that remain in the population after reproduction. Assuming this worst case scenario does not affect generational EAs, because the entire population is replaced by the offspring in each generation, and n fitness values must be always computed — where n is the population size —, independently of the premises.

For an elitist EA, assuming that changes are very hard to detect means that old individuals must be reevaluated and that the average ratio between new solutions and function evaluations, in each generation, is below **1**. In the particular case of ADMGA, it has been shown in [13] that this ratio is approximately $\frac{1}{2}$, meaning that, ADMGA generates only half of the solutions that a standard generational GA is able to generate in the same period of time. This may be particularly penalizing when the frequency of changes is high, and, in fact, ADMGA's performance has been shown to degrade in those situations. The question is: is it possible to improve ADMGA's performance in fast dynamic problems by changing the replacement strategy in a way that those reevaluations are accompanied by the introduction of new genetic material in the population? To assess this hypothesis, three alternative replacement strategies are proposed. Before discussing them, let us describe the main algorithm.

¹ We call *replacement strategy* to the procedure that, from the population of parents $P(t)$ and the population of offspring $P'(t)$, selects the individuals that form the population $P(t+1)$ and then replace population $P(t)$.

Algorithm. ADMGA

```

initialize population  $\mathbf{P}$  with  $size(\mathbf{P}) = n$ 
evaluate population  $\mathbf{P}$ 
set initial threshold  $ts(0)$ 
while (not termination condition)
  create new individuals  $\mathbf{P}'(t)$ 
  evaluate new individuals  $\mathbf{P}'(t)$  //  $n'$  is the size of  $\mathbf{P}'(t)$ 
  if (stationary)  $\mathbf{P}(t+1) \leftarrow \text{best}[\mathbf{P}'(t)+\mathbf{P}(t)]$  //insert the best  $n$  from  $\mathbf{P}'(t)+\mathbf{P}(t)$  into  $\mathbf{P}(t+1)$ 
  if (non-stationary)  $\mathbf{P}(t+1) \leftarrow \mathbf{P}'(t)+\text{best}[\mathbf{P}(t)]$  //insert  $n'$  of  $\mathbf{P}'(t)$  and best  $n - n'$  of  $\mathbf{P}(t)$ 
end while

Procedure: create new individuals
matingEvents  $\leftarrow n/2$ ; successfulMating  $\leftarrow 0$ ; failedMating  $\leftarrow 0$ 
while (successfulMatings  $< 1$ ) do
  for ( $i \leftarrow 1$  to matingEvents) do
    select two chromosomes ( $c_1, c_2$ )
    compute Hamming distance  $H(c_1, c_2)$ 
    if ( $H(c_1, c_2) \geq ts(t)$ )
      crossover and mutate
      successfulMating  $\leftarrow$  successfulMating+1
    end if
    if ( $H(c_1, c_2) < ts(t)$ ) failedMating  $\leftarrow$  failedMating+1
  end for
  if (failedMating  $>$  successfulMating)  $ts(t+1) \leftarrow ts(t) - 1$ 
  else  $ts(t+1) \leftarrow ts(t) + 1$ 
end while

```

Fig. 1. Pseudo-code of the Adaptive Dissortative Mating Genetic Algorithm (ADMGA)**3.1 ADMGA**

ADMGA is a self-regulated dissortative mating GA, which incorporates an adaptive Hamming distance mating restriction that tends to relax as the search process advances. After two parents are selected, crossover only occurs if the Hamming distance between them is found to be above a threshold value. If not, the recombination event is classified as *failed* and another pair of individuals is selected until $n/2$ pair have tried to recombine (n is the population size).

After the reproduction cycle is completed, a new population is created by selecting the n members amongst the parents and newly generated offspring. Then, the threshold is incremented when the number of successful matings is greater or equal than the number of failed matings, and it is decremented otherwise (see pseudo-code in Fig. 1). This way, the genetic diversity indirectly controls the threshold value. When diversity is decreased, threshold tends to be decremented because the frequency of unsuccessful mating will necessarily increase. However, mutation introduces variability in the population, resulting in occasional increments of the threshold that moves it away from 0. The only parameters that need to be tuned in ADMGA is population size n and mutation probability p_m . Crossover probability is not used (in a way, p_c is somewhat adaptive, because selected individuals recombine or not depending on their Hamming distance and the threshold value). As for the threshold, ADMGA has shown to be capable of self-adapting its value in the first generation, and therefore threshold may be set to its highest possible value ($l - 1$, where l is the

chromosome length) in the beginning of the run. However, in order to avoid initial generations in which the ratio between new individuals and function evaluations is very low, an initial threshold value of $l/4$ is used when optimizing non-stationary functions.

RS 1
 insert best $n - n'$ individuals from $\mathbf{P}(t)$ into $\mathbf{P}'(t)$
 $\mathbf{P}(t+1) \leftarrow \mathbf{P}'(t)$ // n is the size of $\mathbf{P}(t)$ and n' is the size of $\mathbf{P}'(t)$

RS 2
 insert mutated best $n - n'$ individuals from $\mathbf{P}(t)$ into $\mathbf{P}'(t)$
 $\mathbf{P}(t+1) \leftarrow \mathbf{P}'(t)$

RS 3
 insert $n - n'$ copies of mutated best from $\mathbf{P}(t)$ into $\mathbf{P}'(t)$
 $\mathbf{P}(t+1) \leftarrow \mathbf{P}'(t)$

RS 4
 insert $n - n'$ random solutions into $\mathbf{P}'(t)$
 $\mathbf{P}(t+1) \leftarrow \mathbf{P}'(t)$

Fig. 2. ADMGA's replacement strategies (RS)

3.2 Replacement Strategies

In order to tackle dynamic optimization problems, the original ADMGA's replacement strategy, proposed in [10], has been already modified in order to reduce the size of the parents' subpopulation that is introduced in the new population. While for stationary functions the algorithm inserts the best n of the complete pool of parents and children into the new population, the ADMGA for non-stationary fitness functions inserts all the n' children in the new population, and then selects the $n - n'$ parents (where n is the population size) to complete the population of solutions.

The modified ADMGA was then tested in dynamic optimization problems, and compared with a standard GA, a standard population-wide elitist GA, RIGA, EIGA and the *Self-Organized Criticality RIGA* (SORIGA) [21] on several problems and dynamics [23]. However, when the frequency of changes is high, ADMGA's performance when compared to the other algorithms diminishes. In order to overcome this difficulty, three different replacement strategies are introduced. Fig. 2 describes the new strategies as well as the original scheme used for dynamic optimization (RS 1). Please note that every strategy inserts the offspring into the new population. The differences reside in the way in which the remaining slots are occupied (i.e., $n - n'$ slots, where n is the population size and n' is the offspring population size).

Replacement strategy 1 (RS 1) — original ADMGA's strategy — inserts the $n - n'$ best individuals from the parents' population into the new population. Replacement strategy 2 (RS 2) fills up the remaining slots with mutated copies of the $n - n'$ best individuals in parents' population (with mutation probability p_m). Replacement strategy 3 (RS 3) inserts $n - n'$ mutated copies of the best solution. Finally, strategy 4 (RS 4) inserts random immigrants — i.e., randomly generated genotypes — into the vacant slots. The following section describes the problems used to test the efficiency of the algorithms.

4 Experimental Setup

The experiments were conducted with dynamic versions of the order-3 trap function, the onemax problem and the the 0 – 1 knapsack problem. With these base-functions, the test set comprises a simple linear functions (onemax), a quasi-deceptive trap functions (order-3 trap) and combinatorial problems (knapsack). These stationary functions were used to construct non-stationary problems with the dynamic problem generator proposed in [22]. This section describes the stationary functions, the dynamic problem generator, and the methodology followed during the experiments.

4.1 Functions

The 0 – 1 knapsack is a class of NP-complete problems that consist in maximizing a *profit* under certain restrictions. Given a set of m items with different weights (\vec{w}) and profits (\vec{p}), the profit $p(\vec{x})$ must be maximized:

$$p(\vec{x}) = \sum_{i=1}^m p_i x_i \quad (1)$$

where $\vec{x} = (x_1 \dots x_m)$ and x_i is 0 or 1, depending if the item is selected ($x_i = 1$) or not ($x_i = 0$). The function is subject to a constraint:

$$\sum_{i=1}^m w_i x_i \leq 0.6 \times \sum_{i=1}^m w_i \quad (2)$$

where w_i is a random integer in the interval [1,50] and p_i is the sum of w_i with a random integer in the interval [1,5]. The knapsack problem used in this study consists of 100 items (meaning that the length of the chromosomes is $l = 100$) with strongly correlated sets of randomly generated data (as in [22]). The fitness of the global optimum is 1853 (since the weights are non-negative integers the global optimum can be obtained with dynamic programming).

A trap function [7] is a piecewise-linear function defined on *unitation* (the number of ones in a binary string) that has two distinct regions in the search space, one leading to a global optimum and the other leading to the local optimum. Depending on its parameters, trap functions may be deceptive or not. The traps in this study are defined by:

$$F(u(\vec{x})) = \begin{cases} k, & \text{if } u(\vec{x}) = k \\ k - 1 - u(\vec{x}), & \text{otherwise} \end{cases} \quad (3)$$

where $u(\vec{x})$ is the unitation function and k is the problem size (and also the fitness of the global optimum). With this equation, order-3 traps are in the region between deceptive and non-deceptive. For this study, a 30 bit problem was constructed by concatenating 10 order-3 subproblems. The fitness of the global optimum is 30.

Finally, the onemax is a simple linear problem that consists in maximising the number of ones in a binary string. For these experiments, we used a 100-bit problem.

4.2 Problem Generator and Methodology

The test environment proposed in [22] was then used to create a dynamic experimental setup based on the functions described above. This problem generator has two parameters that control the severity of the changes and their frequency: ρ is a value between 0 and 1.0 which controls the severity and τ defines number of generations between changes. By changing ρ and τ it is possible to control two of the most important features when testing algorithms on dynamic optimization problems: severity (ρ) and period (τ) — i.e., $1/\tau$ is the frequency — between changes [1]. However, the τ value, if given without the population size n , does not provide enough information on the real period between changes. Therefore, in this paper we use the number of evaluations between each change ε , defined by $\varepsilon = \tau \times n$. For each one of the stationary problems, five different dynamic scenarios were constructed by setting ε to 600, 1200, 2400, 4800, 9600, 19200 and 38400. As for the severity (ρ) value, it is randomly generated in each time the function changes. The scope of this investigation is the performance according to the frequency of changes, and therefore setting ρ to random values simplifies the analysis. However, a final test is made with ε set to 1200 and 4800 and ρ set to 0.05, 0.3, 0.6 and 0.95. Every run covers 50 periods of change, i.e., $50 \times \varepsilon$ evaluations, with changes every ε evaluations.

In order to evaluate an algorithm's configuration when solving a specific problem, the *offline performance* [21] — i.e., the best-of-generation fitness values averaged over the total number of runs and over the data gathering period — is first examined:

$$\overline{F_{BG}} = \frac{1}{G} \times \sum_{i=1}^G \left(\frac{1}{R} \times \sum_{j=1}^R F_{BGij} \right) \quad (4)$$

where G is the number of generations, R is the number of runs (30 in all the experiments) and F_{BGij} is the best-of-generation fitness of generation i of run j of an algorithm on a specific problem. This value gives information on how close the GAs are able to track the moving solution.

A GA has several parameters that model their general behavior. We are particularly interested in GAs' performance when varying the mutation probability, because evolutionary approaches that work by maintaining population diversity at a higher level during the search may be shifting the optimal mutation probability to different values. In addition, it has been demonstrated [17] that dissortative and assortative mating increase and decrease, respectively, the optimal mutation probability of a GA. Therefore, it is of extreme importance to test the GAs under a reasonable range of p_m values, otherwise the results may become biased toward some of the approaches. Probability values p_m were set to $1/(2 \times l)$, $1/l$, $2/l$ and $4/l$.

The population size n also affects the performance of the GAs, not only on static problems, but also in dynamic environments. Knowing the optimal size is important for determining with accuracy the scalability of a GA and to avoid superfluous computation effort due to a population larger than the optimal. Although this investigation does not aim at studying scalability, a proper research methodology must test different n values, otherwise there is a risk of comparing suboptimal parameter settings and, consequently, getting invalid conclusions. In this study, all the algorithms were tested with $n = 8, 16, 30, 60$ and 120 . Uniform crossover was chosen in order to avoid taking advantage of the trap function building blocks tight linkage.

Every algorithm in the test set uses binary tournament (tournament size **2** is in general a fairly good selective pressure for most problem [20]).

The ADMGA was compared with GGA and EIGA. EIGA is a very simple scheme that in each generation replaces a fraction r_i of the population by mutated copies of the best solution of the previous generation (with mutation probability p_m^i). In [23], Yang shows that the algorithm is more effective when the changes are not too severe. Due to its simplicity and the interesting results reported in [23], EIGA was selected as the main peer-algorithm for this study. In addition, EIGA has some similarities with one of the replacement strategies proposed in this paper to improve ADMGA's performance, which makes in a suitable candidate for being included in the test set.

RS 4 was found to be the worst replacement strategy in the test set, being unable to deal with the proposed dynamic problems. RS 4 is not a proper strategy for ADMGA and therefore, in order to simplify the graphics, it was removed from analysis and discussion in Section 5.

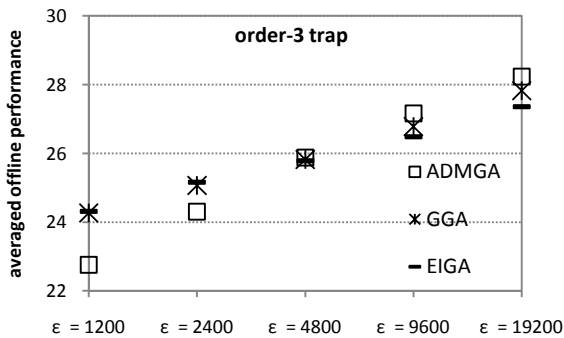


Fig. 3. Order-3 dynamic trap function. Standard ADMGA, GGA and EIGA. Population size $n = 30$; $p_m = 1/l$ (GGA), $p_m = 2/l$ (EIGA and ADMGA); $p_c = 1.0$; GGA with 2-elitism.

GGA was tested with crossover probability set to 0.7 and 1.0, with and without elitism. The best results are attained with $p_c = 1.0$ and 2-elitism. Like the other algorithms, EIGA was also tested with several p_m values; p_c was set to 0.6 (as suggested in [23]), 0.7 and 1.0; r_i is set 0.2 (also, as suggested in [23]). Please note that due to its design, EIGA population size n^* must set so that $(1 + r_i) \times n^* = n$, where n is the population size of a standard GA that would perform the same number of function evaluations in each generation. EIGA was tested with different n^* values and the results discussed in the following section refer always to the best configurations. Please refer to [23] for details the algorithm's implementation and parameter tuning.

5 Results and Discussion

Fig. 3 illustrates the issue addressed by this study. ADMGA only outperforms the other GAs when ϵ is above a specific value. In the results on depicted in the graphic,

ADMGA is clearly outperformed by the other algorithms when $\varepsilon < 4800$ (confirmed by statistical tests). The main objective is to find a replacement strategy for ADMGA that reduces this value.

Fig. 4 summarizes the results attained by the different versions of ADMGA. The graphics show the configurations with n and p_m values that maximize the performance of each replacement strategy. These results demonstrate RS 2 is capable of outperforming standard ADMGA (RS 1) in the high frequency scenarios. Replacement strategy 3, which introduces mutated copies of the best individual in the population, works well in the onemax problem, but it is outperformed by the other strategies in most of the dynamic scenarios based knapsack and trap function. (RS 2 is 2-elitist, because it improves its performance. Please note that RS 2 is quite disruptive, but the elitism guarantees that the best solutions are not lost.)

Table 1 summarizes the statistical tests. RS 2 is compared with RS 1 using Kolmogorov-Smirnov tests with 0.05 level of significance. The tests show that RS 2 clearly outperforms standard ADMGA (RS 1) in most of the problems. The first objective of this study has been accomplished: one of the schemes is able to improve ADMGA's performance in fast dynamic problems. The following step is to compare the new ADMGA with other GAs.

As already stated, GGA and EIGA were thoroughly tested in order to avoid unfair comparisons. GGA is better with $p_c = 1.0$ and 2-elitism. Best population size is $n = 16$ for onemax, and $n = 30$ for order-3 trap and knapsack. In general, GGA's performance is optimized by $p_m = 1/l$ except on knapsack, for which the best p_m is $2/l$. Fig. 5 and Table 2 shows that for $\varepsilon > 2400$, ADMGA with RS 2 is never outperformed by GGA. In particular, the ε value above which ADMGA is at least equivalent to GGA decreases from 4800 to 600 in order-3 trap (please compare Fig. 3 and Fig. 5). Table 3 compares ADMGA with the standard strategy (RS 1) and GGA. By comparing the results in Table 2 and Table 3, it is perceptible that RS 2 reduces the ε above which ADMGA is significantly better or at least statistically equivalent to GGA in the three types of dynamic fitness functions. If we compare ADMGA's replacement strategy 2 with EIGA the conclusions are similar — see Fig. 5 and Table 4. EIGA performs better than ADMGA (RS 2) in fast onemax problem and knapsack problems. On the other hand, EIGA is outperformed by ADMGA in most of the dynamic order-3 trap problems.

As stated above, the comparisons in this study were made considering the worst-case scenario, i.e., changes are hard to detect and a reliable detection requires the reevaluation of the chromosomes that are copied from previous generations. However, we may consider a different assumption: changes are easy to detect and all that is required is to reevaluate old chromosomes after a change is detected. Under these conditions, the results are different. A summary of EIGA and ADMGA's behavior is shown in Table 5: ADMGA clearly outperforms EIGA in almost every dynamic problem. However, at this point we cannot exclude the possibility that population-wide elitism may be biasing the results towards ADMGA under these conditions; therefore, other experiments must be devised in order to compare properly the GAs.

A final test was conducted with dynamic scenarios with fixed severity values. ADMGA with RS 1 and RS 2 was tested and compared on dynamic problems with ε set to 1200 and 4800 and severity ρ set to 0.05, 0.3, 0.6 and 0.95. The algorithms were tested with the parameter values that maximize their performance on the random severity problems.

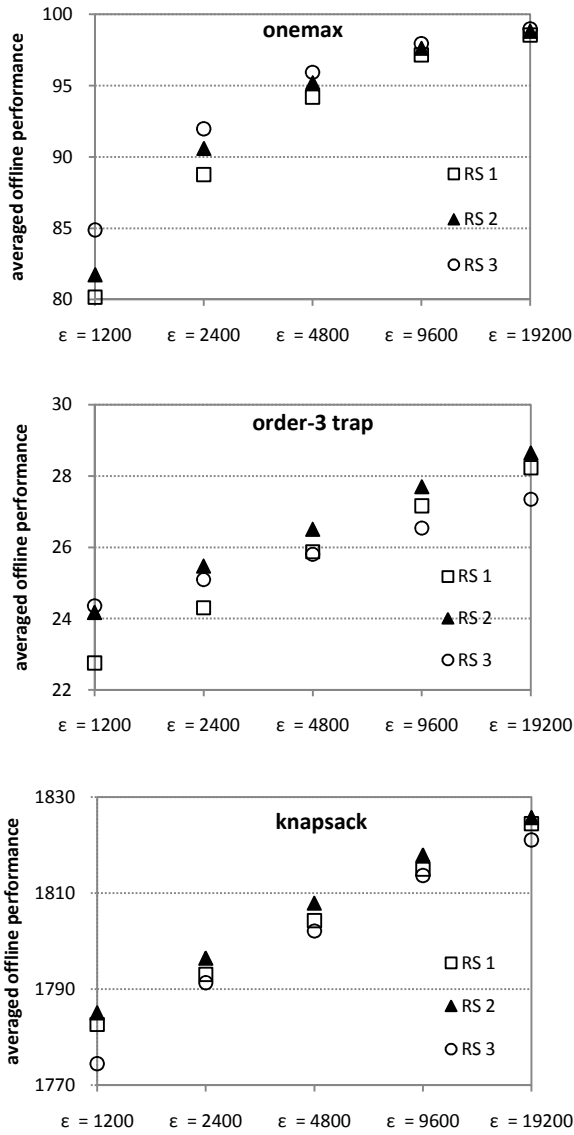


Fig. 4. ADMGA replacement strategies. Onemax, order-3 trap and knapsack dynamic problems. Population size: $n = 16$ (onemax) and $n = 30$ (trap and knapsack); $pm = 2/l$ (RS 1) and $pm = 1/l$ (RS 2 and RS 3). RS 2 with 2-elitism.

Table 1. Kolmogorov-Smirnov tests (RS 2 vs RS1). Results are shown as + signs when ADMGA (RS 2) is significantly better than the ADMGA (RS 1), – when RS 2 is significantly worst, and \approx when the differences are not statistically significant. Parameters as in Fig. 4.

$\epsilon \rightarrow$	600	1200	2400	4800	9600	19200	38400
onemax	+	+	+	+	\approx	\approx	\approx
trap	+	+	+	+	+	+	+
knapsack	+	+	+	+	+	+	+

Table 2. Kolmogorov-Smirnov tests (RS 2 vs GGA). The results are shown as + signs when ADMGA with RS 2 is significantly better than GGA, – when RS 2 is significantly worst, and \approx when the differences are not statistically significant. Parameters as in figure 5.

$\epsilon \rightarrow$	600	1200	2400	4800	9600	19200	38400
onemax	–	–	\approx	\approx	\approx	\approx	\approx
trap	\approx	\approx	+	+	+	+	+
knapsack	–	–	\approx	\approx	\approx	+	+

Table 3. Kolmogorov-Smirnov tests (RS 1 vs GGA). The results are shown as + signs when ADMGA with RS 1 is significantly better than GGA, – when RS 1 is significantly worst, and \approx when the differences are not statistically significant. Parameters as in figures 4 and 5.

$\epsilon \rightarrow$	600	1200	2400	4800	9600	19200	38400
onemax	–	–	–	–	\approx	\approx	\approx
trap	–	–	–	\approx	+	+	+
knapsack	–	–	–	–	–	\approx	+

Table 4. Kolmogorov-Smirnov tests (RS 2 vs EIGA). The results of the test are shown as + signs when ADMGA with RS 2 is significantly better than EIGA, – when RS 2 is significantly worst, and \approx when the differences are not statistically significant. Parameters as in figure 5.

$\epsilon \rightarrow$	600	1200	2400	4800	9600	19200	38400
onemax	–	–	–	\approx	\approx	\approx	\approx
trap	\approx	\approx	+	+	+	+	+
knapsack	–	–	\approx	\approx	\approx	\approx	\approx

Table 5. Kolmogorov-Smirnov tests (RS 2 vs EIGA). The results of the test are shown as + signs when ADMGA with RS 2 is significantly better than EIGA, – when RS 2 is significantly worst, and \approx when the differences are not statistically significant. Parameters as in Fig. 5.

$\epsilon \rightarrow$	600	1200	2400	4800	9600	19200	38400
onemax	+	+	+	+	+	+	\approx
trap	+	+	+	+	+	+	+
knapsack	+	+	+	+	+	+	+

Finally, a test with fixed severity shows that the new replacement strategy is able to improve the standard strategy performance on a wide range of severity values.

Table 6 shows the numerical results and the statistical tests. RS 2 improves the performance of the standard strategy in most of the scenarios. These results validate the approach and show that it is possible to improve ADMGA's performance on dynamic optimization problems by adjusting the replacement strategy.

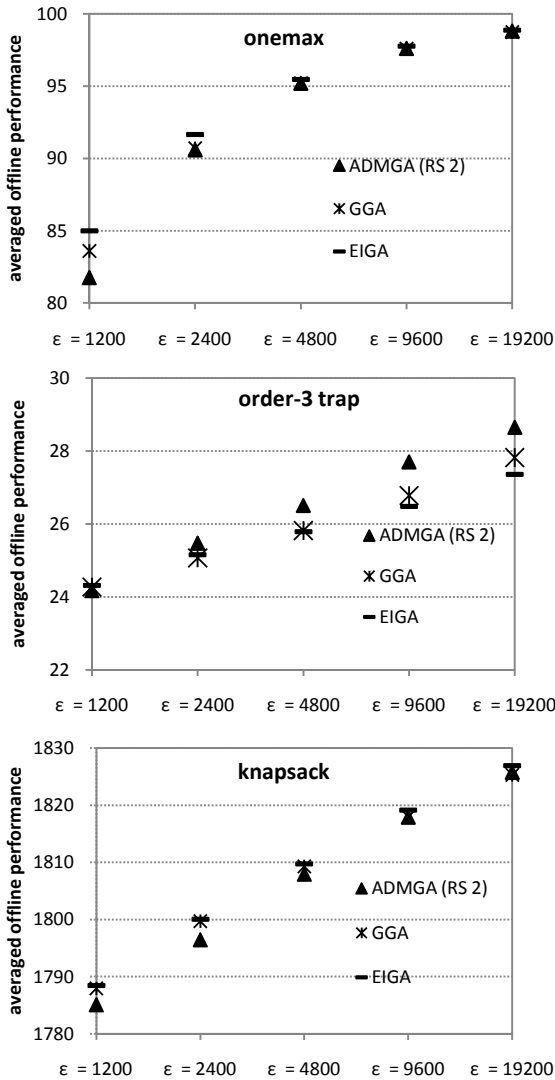


Fig. 5. ADMGA (RS 2), GGA and EIGA. Parameters as in fig. 3 and 4. Population size $n = 16$ (onemax) and $n = 30$ (order-3 and knapsack). GGA with $p_m = 1/l$ (onemax and trap) and $p_m = 2/l$ (knapsack). EIGA with $p_m = 2/l$ and $r_i = 0.2$.

Table 6. Numerical results and Kolmogorov tests (R1 vs R2). Results are shown as + signs when ADMGA (RS 2) is significantly better than the ADMGA (RS 1), - when RS 2 is significantly worst, and \approx when the differences are not statistically significant. Population size $n = 30$. Trap: ADMGA (RS 1): $p_m = 1/l$; ADMGA (RS 2): $p_m = 1/(2 \times l)$. Knapsack: ADMGA (RS 1): $p_m = 4/l$ ($\epsilon = 1200$) and $p_m = 2/l$ ($\epsilon = 4800$). ADMGA (RS 2): $p_m = 4/l$ ($\epsilon = 1200$) and $p_m = 2/l$ ($\epsilon = 4800$).

$\rho \rightarrow$		$\epsilon = 1200$				$\epsilon = 4800$			
		0.05	0.3	0.6	0.95	0.05	0.3	0.6	0.95
trap	RS 1	28.35 ± 0.40	23.96 ± 0.26	22.66 ± 0.21	24.68 ± 0.08	29.46 ± 0.13	27.35 ± 0.21	25.53 ± 0.16	25.13 ± 0.05
	RS 2	28.74 ± 0.19 (+)	24.25 ± 0.28 (+)	22.92 ± 0.20 (+)	24.73 ± 0.07 (+)	29.73 ± 0.04 (+)	28.10 ± 0.10 (+)	25.90 ± 0.13 (+)	25.26 ± 0.07 (+)
knapsack	RS 1	1807.41 ± 1.35	1794.94 ± 1.16	1785.57 ± 0.32	1767.99 ± 1.25	1827.30 ± 0.99	1811.84 ± 0.70	1800.84 ± 0.98	1789.27 ± 0.56
	RS 2	1800.76 ± 0.91 (-)	1796.52 ± 0.83 (+)	1791.09 ± 0.88 (+)	1779.46 ± 0.86 (+)	1819.90 ± 0.65 (-)	1812.29 ± 0.56 (+)	1804.79 ± 0.57 (+)	1794.88 ± 0.42 (+)

6 Conclusions

This paper proposes new replacement schemes for the Adaptive Dissortative Mating Genetic Algorithm (ADMGA). The main objective is to improve standard ADMGA’s performance in dynamic problems with high frequency of changes. One of the proposed strategies outperforms the standard strategy in most of the dynamic scenarios designed for testing the algorithms. The new strategy simply mutates the chromosomes that remain in the population after the recombination stage before reevaluating them.

The results show that ADMGA is capable of outperforming not only a standard GA, but also the *Elitism-based Immigrants* GA (EIGA) in some classes of problems and dynamics with random severity: 1) when the frequency of changes is lower, ADMGA is never outperformed by the other GAs; 2) with higher frequencies, ADMGA is never outperformed by GGA and EIGA in order-3 trap functions. Preliminary tests in non-stationary functions in which the changes are easy to detect, show that ADMGA is able to outperform EIGA in all but one scenario. Finally, a test with fixed severity shows that the new replacement strategy is able to improve the standard strategy performance on a wide range of severity values.

One of ADMGA’s advantages over other GAs is that it only requires two parameters that need to be tuned (n and p_m), while EIGA, for instance, requires the setting of four parameters (n , p_m , p_c and ri). Since EIGA has been recently proposed as a GA specifically conceived for dynamic optimization, and since the report in [16] claims that the algorithm performs well on dynamic optimization problems, we may state that ADMGA is a viable strategy for tackling dynamic optimization problems.

Acknowledgements. The first author wishes to thank FCT, *Ministério da Ciência e Tecnologia*, his Research Fellowship SFRH / BPD / 66876 / 2009, also supported by FCT (ISR/IST plurianual funding) through the PIDDAC Program Funds. The paper has also been funded in part by the Junta de Andalucía P06-TIC-02025 and P07-TIC-03044.

References

1. Angeline, P.: Tracking Extrema in Dynamic Environments. In: Proceedings of the 6th International Conf. on Evolutionary Programming, pp. 335–345. Springer (1997)
2. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press (1996)
3. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the 1999 IEEE Congress on Evolutionary Computation, pp. 1875–1882. IEEE Press (1999)
4. Branke, J., Kaußler, T., Schmidt, C., Schmeck, H.: A multi-population approach to dynamic optimization problems. In: Parmee, I.C. (ed.) Proceedings of the Adaptive Computing in Design and Manufacturing (ACDM 2000), pp. 299–308. Springer, London (2000)
5. Branke, J.: Evolutionary optimization in dynamic environments. Kluwer, Norwel (2001)
6. Cobb, H. G.: An investigation into the use of hypermutation as an adaptive operator in GAs having continuous, time-dependent nonstationary environments. Technical Report AIC-90-001. Naval Research Laboratory, Washington (1990)
7. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In: Whitley, L.D. (ed.) Foundations of Genetic Algorithms 2, pp. 93–108. Morgan Kaufmann, San Francisco (1993)
8. Eschelman, L.J., Schaffer, J.D.: Preventing premature convergence in genetic algorithms by preventing incest. In: Proceedings of the 4th International Conference on Genetic Algorithms, pp. 115–122. Morgan Kauffman, San Francisco (1991)
9. Fernandes, C.M., Rosa, A.C.: A Study on Non-Random Mating in Evolutionary Algorithms Using a Royal Road Function. In: Proceedings of the 2001 Congress on Evolutionary Computation, pp. 60–66. IEEE Press (2001)
10. Fernandes, C.M., Rosa, A.C.: Self-adjusting the intensity of dissortative mating of genetic algorithms. *Journal of Soft Computing* 12, 955–979 (2008)
11. Fernandes, C.M., Rosa, A.C.: Evolutionary Algorithms with Dissortative Mating on Static and Dynamic Environments. In: Kosinski, W. (ed.) *Advances in Evolutionary Algorithms*, pp. 181–206. In-Tech (2008)
12. Fernandes, C.M., Merelo, J.J., Ramos, V., Rosa, A.C.: A Self-Organized Criticality Mutation Operator for Dynamic Optimization Problems. In: Keijzer, M. (ed.) *Proceedings of the 2008 Genetic and Evolutionary Computation Conference*, pp. 937–944. ACM Press (2008)
13. Fernandes, C.M.: Diversity-enhanced Genetic Algorithms for dynamic optimization. Ph.D Thesis. Technical University of Lisbon (2009), <http://geneura.ugr.es/pub/tesis/PhD-CFernandes.pdf>
14. Goldberg, D.E., Smith, R.E.: Nonstationary function optimization using genetic algorithms with dominance and diploidy. In: Grefenstette, J.J. (ed.) *Proceedings of the 2nd International Conference on Genetic Algorithms*, pp. 59–68. L.Erlbaum Associates, Hillsdale (1987)

15. Grefenstette, J.J.: Genetic algorithms for changing environments. In: Manner, R., Manderick, B. (eds.) *Parallel Problem Solving from Nature II*, pp. 137–144. North-Holland, Amsterdam (1992)
16. Ochoa, G., Mädler-Kron, C., Rodriguez, R., Jaffe, K.: Assortative Mating in Genetic Algorithms for Dynamic Problems. In: Rothlauf, F., Branke, J., Cagnoni, S., Corne, D.W., Drechsler, R., Jin, Y., Machado, P., Marchiori, E., Romero, J., Smith, G.D., Squillero, G. (eds.) *EvoWorkshops 2005. LNCS*, vol. 3449, pp. 617–622. Springer, Heidelberg (2005)
17. Ochoa, G.: Error Thresholds in Genetic Algorithms. *Evolutionary Computation* 14(2), 157–182 (2006)
18. Ramsey, C.L., Grefenstette, J.J.: Case-based initialization of genetic algorithms. In: *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 84–91. Morgan Kaufmann (1993)
19. Russel, P.J.: *Genetics*. Benjamin/Cummings (1998)
20. Thierens, D.: Scalability problems of simple GAs. *Evolutionary Computation* 7(4), 331–352 (1999)
21. Tinós, R., Yang, S.: A self-organizing random immigrants GA for dynamic optimization problems. *Genetic Programming and Evolvable Machines* 8(3), 255–286 (2007)
22. Yang, S., Yao, X.: Experimental study on PBIL algorithms for dynamic optimization problems. *Journal of Soft Computing* 9(11), 815–834 (2005)
23. Yang, S.: Genetic Algorithms with Memory- and Elitism-Based Immigrants in Dynamic Environments. *Evolutionary Computation* 16(3), 385–416 (2008)

A Receding Horizon Genetic Algorithm for Dynamic Resource Allocation: A Case Study on Optimal Positioning of Tugs

Robin T. Bye

Department of Technology and Nautical Sciences, Ålesund University College
Postboks 1517, N-6025 Ålesund, Norway
roby@hials.no
<http://www.robinbye.com>

Abstract. This paper presents a receding horizon genetic algorithm (RHGA) for dynamic resource allocation. The algorithm combines methods from control theory and computational intelligence to simultaneously solve the problems of (i) coordinated control of resources, (ii) task assignment, and (iii) multiple target tracking in a dynamic environment. A simulated case study on optimal positioning of a fleet of tugs along the northern Norwegian coast serves as a means of evaluating the algorithm. In terms of reducing the risk of oil tanker drifting accidents, the study shows that the RHGA is able to iteratively plan movement trajectories for each individual tug such that the net collective behaviour of the tugs outperforms that of stand-by tugs stationed at bases located uniformly along the coast. The promising results suggest great potential for further development and generalisation to other dynamic resource allocation problems.

Keywords: Dynamic resource allocation, genetic algorithm, receding horizon control, model predictive control, optimal control.

1 Introduction

Dynamic resource allocation can be considered a broad class of optimisation problems, including search and rescue operations, vehicle routing, crew allocation and scheduling, and many others. This study focuses on the problem of allocating a group of resources to the tracking of multiple targets in a dynamic environment. Specifically, it considers a fleet of tugs operating along a coast line with the purpose of preventing oil tankers from drift grounding. The tugs must dynamically be assigned moving target positions for tracking such that the overall risk of any oil tankers drifting aground is minimised. Such a problem is a demanding one and poses a number of interrelated challenges.

A first challenge is that of task assignment: Which resources shall track which targets? On the one hand, if there are more resources than targets, a subset of resources could be given the task of tracking one target each whilst remaining resources could be given the task of self-maintenance or simply doing nothing. On the other hand, if the number of targets exceeds the number of resources, some resources must be assigned more than one target. In both cases, some governing principle is needed for allocating resources.

A second challenge is that of target tracking: How should each resource move to best track, or cover, its assigned targets? When targets outnumber resources, there will be an inherent tradeoff between good tracking of some targets at the expense of bad tracking of others. In addition, some targets may be considered more important to track than others and therefore must be weighted more.

Collectively considering task assignment and target tracking for the resources as a group, a third challenge is that of coordinated control and collective performance: How should tasks be assigned and targets be tracked such that some net performance index is optimised?

Finally, being in a dynamic environment, these challenges need to be constantly reevaluated: How can future changes in the state space, such as motion of targets and changing dynamics of the surroundings, be incorporated?

This paper presents a receding horizon genetic algorithm (RHGA) for solving the abovementioned challenges. The performance of the algorithm is demonstrated in a simulated case study of a real-world problem, namely the positioning of tug vessels along the coastline of northern Norway.

The following sections provide background information on the tug positioning problem, derivation of a simplified and concise problem description, and details on a proposed algorithm that solves the problem by combining receding horizon control (RHC) with a genetic algorithm (GA). The performance of the RHGA will be demonstrated in some simulated scenarios. Finally, the simulation results, aspects of our approach, and future potential will be discussed.

1.1 The Tug Positioning Problem

Each year thousands of ship transits, including several hundred transits of oil tankers, are made along the coastline of northern Norway, thus exposing it to the risk of drift grounding accidents and oil spill [1]. In an effort to reduce the risk of such accidents, the Norwegian Coastal Administration (NCA) runs a vessel traffic services (VTS) centre in the town of Vard, which administers a fleet of tugs patrolling the coastline. The main purpose of these tugs is to cleverly patrol the coastline in such a manner that if an oil tanker loses manoeuvrability through steering or propulsion failure, there will be a tug sufficiently close that it can intercept the drifting oil tanker before it runs ashore [2].

Oil tankers are required by law to sail along predefined piecewise-linear corridors approximately parallel to the coastline. Hence, for example by linearly extrapolating its speed along its corridor, it is possible to predict a tankers future position. Moreover, all ships are required by international law to constantly transmit both static (identity, dimensions, cargo, etc.) and dynamic (position, speed, heading, etc.) ship information through the automatic identification system (AIS). The AIS information is transmitted both to other ships and nearby VTS centres and relayed on the Internet. Together with weather forecasts and dynamic models of wind, wave heights, and ocean currents, the AIS information can be used to predict potential drift trajectories and grounding locations for ships that lose manoeuvrability [2].

The NCA has developed risk-based decision support tools based on dynamical risk models that draws on a vast pool of information [23]. Some of this information is static and certain, such as a the type of ship, the nationality of its crew, and the amount

and type of oil it is carrying. Information about other factors is dynamic and uncertain and requires modelling. Such factors include wind, waves, currents, accident frequency and consequences, oil spill size and potential impact, and others. The decision support tools aid the human operator at a VTS centre in directing tugs by determining high-risk target areas that tugs should approach. Nevertheless, with the projected rapid increase in oil tanker transits in coming years [1] and the increasing number of tugs required for adequate patrolling, the problem quickly becomes unmanageable by a human operator. Consequently, there is a need of an algorithm able to calculate position trajectories that each tug should follow in order to reduce the overall risk of drifting accidents.

1.2 Problem Formulation

Before developing an algorithm, the tug positioning problem must be formulated carefully and precisely. First, it is assumed that N_o oil tankers move in one dimension only (north or south, say) along a line of motion z . This is a reasonable assumption considering that oil tankers follow predefined piecewise-linear corridors. Inside of z and closer to shore, it is assumed that N_p tugs are patrolling along a line of motion y parallel to z . The possibility of collisions between oil tankers and patrol tugs on their respective lines of motion is not considered.

It is acknowledged that the coastline does not constitute a sequence of connected straight line segments due to its vast amount of fjords, peninsulas, and islands. Nevertheless, because tugs should stop drifting ships *before* they reach land or danger zones, a straight patrol line some distance from the rugged coastline can be considered a conservative choice. Figure 1 shows a graphical representation of the problem description, illustrated by two patrolling tugs and three oil tankers.

Moreover, the algorithm assumes real-time access to prediction data from a set of accurate models such as those developed by the NCA and described in Sect. 1.1. These models must be able to predict future positions of oil tankers along z and the corresponding potential drift trajectories given current and predicted information about the tankers themselves and the environment they are operating in.

Suppose an oil tanker currently positioned at $z(t)$ starts drifting at some time $t = t_d$. The algorithm requires a future position trajectory predicted T_h hours ahead in time, where T_h is called the prediction horizon. Employing a discrete-time model with a sampling period of $T_s = 1$ hour, the estimated future positions are given by $\hat{z}(t|t_d)$ for $t = t_d + 1, t_d + 2, \dots, t_d + T_h$.

Moreover, for each predicted position $\hat{z}(t|t_d)$ there is a corresponding predicted drift trajectory starting at $\hat{z}(t|t_d)$ that may or may not intersect the patrol line y after an estimated drift time $\hat{\Delta}$ into the future depending on ocean currents, wave heights, wind conditions, oil tanker shape and weight, and more. Collecting all predicted drift trajectories for all oil tankers results in a distribution of *cross points* where future drift trajectories will intersect the patrol line.

Based on the predicted distribution of cross points, the problem is to calculate trajectories, or sequences of *patrol points*, along y for each of the patrolling tugs such that the risk of an oil tanker in drift not being reached and towed to safety before grounding is minimised. This is a difficult problem involving collective behaviour, task assignment, and multi-target tracking in a dynamic environment.

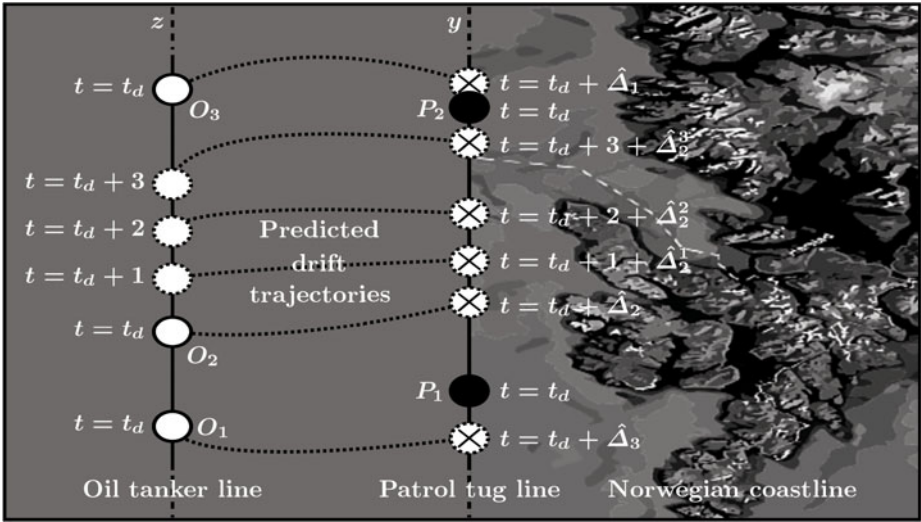


Fig. 1. Problem description. Patrol tugs P_1 and P_2 (black circles) and oil tankers O_1 , O_2 , and O_3 (white circles) move unidimensionally along lines y and z , respectively. Solid circles correspond to positions at current time of drift $t = t_d$, whereas dashed lines and circles indicate predicted drift trajectories and positions for $t > t_d$. The estimated duration of each drift trajectory is denoted $\hat{\Delta}_i^k$, where i and k refers to the drift trajectory for the i th oil tanker that begins k hours ahead in time at $t = t_d + k$. Circles with a cross indicate cross point points. The dynamic resource allocation problem is that of determining how the tugs collectively should move in the time ahead in order to best reduce the risk of drift grounding accidents.

2 Method

An example scenario in Fig. 2 shows three tugs and three corresponding random walk patrol trajectories the tugs may follow in order to track the cross point distributions of six oil tankers. As seen from the plot, these patrol trajectories provide poor coverage of cross points. The tugs stay more or less around their initial positions, leaving a large number of cross points unattended. A better solution would have the tugs spread out, each tug covering its own set of cross points, and thus improving the overall coverage.

How can the quality, or performance, of the choice of patrolling tug trajectories be measured? One possible approach is to examine a large number of sets of potential patrol trajectories and for each set evaluate a cost function that quantifies the performance of the tugs. There are several methods that likely can find near-optimal solutions in reasonable time for this approach, for example variants of Monte Carlo methods, simulated annealing, ant colony optimisation, genetic algorithms (GAs), or other methods from computational intelligence (e.g., see [4] for an overview). This study uses a continuous GA based on a version similar to that described in [5].

Another challenge is how to accommodate changing dynamics of the environment. Not only may oil tankers change their speeds and headings but weather conditions also constantly change. Consequently, cross point distributions will change with time and

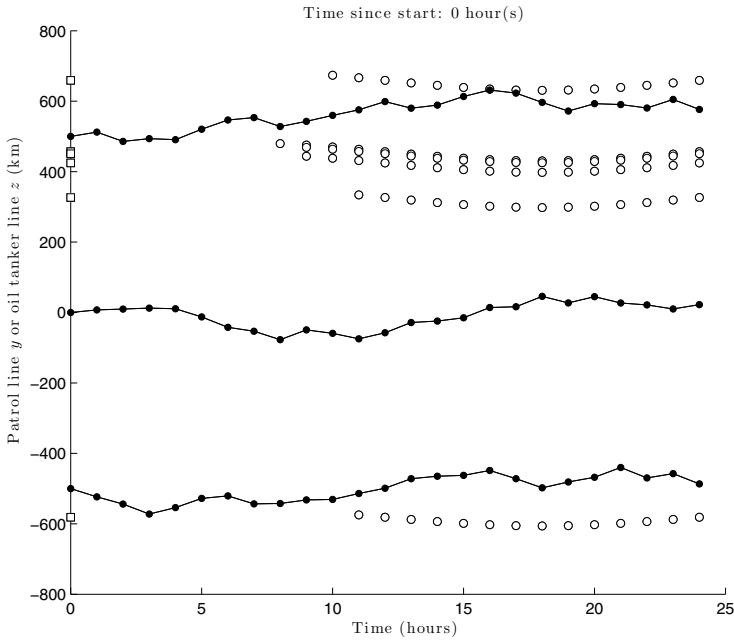


Fig. 2. Example scenario. White squares on the vertical axis indicate the initial positions of six oil tankers on the oil tanker line z at $t = 0$. White circles indicate where the corresponding predicted drift trajectories will cross the patrol line y as a function of time. Black circles indicate three random walk patrol trajectories on y for $t = 0, 1, \dots, T_h$, where the prediction horizon is set to $T_h = 24$ hours. The area reachable by each of the three tugs at full speed from their initial positions are depicted by envelopes delimited by dotted lines. Note that the smallest observed drift time is $\hat{\Delta} = 8$ hours, whereas the largest drift time is $\hat{\Delta} = 11$ hours. An algorithm for tug positioning must generate patrol trajectories such that the tugs cover the distribution of cross points as well as possible.

solutions must be recalculated. One solution is to redo the search for candidate patrol trajectories at regular intervals and replace the previously generated trajectories with new ones. Such a scheme is implemented in the tug positioning algorithm by utilising receding horizon control (RHC).

Details of the GA and RHC strategy are presented in Sects. [2.1](#) and [2.2](#) respectively.

2.1 The Genetic Algorithm

A GA is a heuristic search method based on principles of natural evolution (e.g., see [\[6,5,4\]](#) for detailed descriptions). GAs are particularly useful for obtaining solutions to difficult optimisation and search problems where the solution space is nonconvex. Although no mathematical analysis about the convexity of the tug position problem is provided in this paper, it seems clear that generating patrol trajectories that are optimal, or near-optimal, in some sense by minimising a cost function such as [\(3\)](#) (defined later) is not easy and utilising a GA is an appropriate choice.

Characteristics of the GA. The GA used here consists of the following steps, which adheres to the general scheme used in most GAs (e.g., see [5]):

1. Define a cost function and a chromosome encoding and set some GA parameters such as mutation and selection.
2. Generate an initial population of chromosomes.
3. Evaluate a cost for each chromosome.
4. Select mates based on a selection parameter.
5. Perform mating.
6. Perform mutation based on a mutation parameter.
7. If the desired number of iterations or cost level is reached, stop algorithm and return solution, otherwise, repeat from Step 3.

The selection parameter is in the range 0–1 and determines how many chromosomes in a population survives from one iteration to the next. The cost associated with each chromosome is evaluated and the chromosomes are given a weighted selection probability according to their cost, where a smaller cost results in a greater probability. For a selection parameter of 0.5, half the population is then randomly picked, with low cost chromosomes having a greater chance of being picked and kept for survival and reproduction. The other chromosomes are discarded to make room for new offspring.

For mating, the GA uses a combination of an extrapolation method and a crossover method. Information from two parent chromosomes are combined with an extrapolating method to obtain new offspring variable values bracketed by the parents' variable values. A single crossover point is used to determine which parts of the parent chromosomes are used for creating offspring.

After mating, a fraction of the genes are mutated, which means that the values of these genes are changed to random numbers within an allowable range. A mutation rate determines how many genes are mutated at every iteration.

Particular to the problem described in this paper is the choice of cost function and chromosome encoding, which are described in the following.

Cost Function. Proper choice of a cost function is imperative for the algorithm to find desirable solutions. Here, the cost function is defined as the sum of the distances between all cross points and the *nearest* patrol points. The rationale behind this choice is that if an oil tanker in drift can be saved by a tug a certain distance away, it is not important that other tugs further away can save it at a later time.

Note that choosing distance as a cost measure is equivalent to minimum rescue time if one assumes that all tugs have the same maximum speed. For cases where tugs have different maximum speeds, one could define rescue time as distance divided by maximum tug speed and sum the minimum rescue times for each cross point.

By intuition, the emphasis on punishing distances to only the nearest patrol points should yield proper task assignment, as good solutions found by the GA will tend to have patrol tugs spreading out and tracking different groups of cross points, thus collectively reducing the overall risk of grounding. Indeed, this intuition is confirmed by the results presented in Sect. 3.

A cross point (position on y intersected by a drift trajectory) of the c th oil tanker's drift trajectory at time t can be defined as y_t^c . For the prediction horizon T_h there is a set of cross points given by

$$\{y_t^c\} = \{y_{t_d}^c, y_{t_d+1}^c, \dots, y_{t_d+T_h}^c\} , \quad (1)$$

however, only a subset of these points are defined, since the drift trajectories must actually cross the patrol line at the specified times. For example, if the predicted drift time is $\hat{\Delta}$ for all drift trajectories, the earliest occurrence of a cross point of a drift trajectory starting on z at $t = t_d$ will be at $t = t_d + \hat{\Delta}$, and

$$\{y_t^c\} = \{y_{t_d+\hat{\Delta}}^c, y_{t_d+1+\hat{\Delta}}^c, \dots, y_{t_d+T_h}^c\} . \quad (2)$$

A patrol point (tug position on y) on the p th tug's patrol trajectory at time t can be defined as y_t^p . For N_o oil tankers and N_p patrol tugs, then, the cost $f(t, \mathbf{C}_i)$ is defined as a function of time t and the i th chromosome \mathbf{C}_i by

$$f(t, \mathbf{C}_i) = \sum_{t=t_d}^{t_d+T_h} \sum_{c=1}^{N_o} \min_{p \in P} |y_t^c - y_t^p| , \quad (3)$$

where $P = \{1, 2, \dots, N_p\}$ and details on y_t^p and \mathbf{C}_i are given below.

Chromosome Encoding. For tug p , consider a sequence $\{u_t^p\}$ consisting of T_h normalised control inputs, or speed commands, u_t^p , where

$$\{u_t^p\} = \{u_{t_d+1}^p, u_{t_d+2}^p, \dots, u_{t_d+T_h}^p\} , \quad -1 \leq u_t^p \leq 1 . \quad (4)$$

The maximum control input values of -1 and 1 are equivalent to tugs going with maximum speed in the negative or positive y -direction, respectively. This encoding is generic as it is independent of each tug's maximum speed.

Given a control input u_t^p , a point y_t^p on the patrol trajectory for tug p at time t can be obtained through linear extrapolation using the difference equation

$$y_t^p = y_{t-1}^p + u_t^p v_{\max}^p T_s , \quad (5)$$

where v_{\max}^p is the maximum speed for the p th tug and T_s is the duration of each time step. The entire patrol trajectory is a sequence of T_h patrol points given by

$$\{y_t^p\} = \{y_{t_d+1}^p, y_{t_d+2}^p, \dots, y_{t_d+T_h}^p\} . \quad (6)$$

To encode N_p control trajectories as sequences $\{u_t^p\}$ of length T_h for each patrol tug $p \in P$, the i th chromosome \mathbf{C}_i of length $N_p \times T_h$ is encoded as

$$\mathbf{C}_i = \left\{ u_1^1, \dots, u_{T_h}^1, u_1^2, \dots, u_{T_h}^2, \dots, u_1^{N_p}, \dots, u_{T_h}^{N_p} \right\} . \quad (7)$$

That is, each chromosome is a concatenation of N_p control trajectories, each of which consists of T_h future control inputs. Given an initial tug position $y_{t_d}^p$ and employing (5) repeatedly, these control trajectories are used to generate the patrol trajectories in (6).

2.2 Receding Horizon Control

Because of the dynamics of the problem, where neither oil tankers speed and heading nor wind, wave, and ocean current conditions are static, patrol trajectories optimised by the GA will soon become outdated. One possibility is to run the GA at regular intervals, constantly incorporating updated *current* information about the state of the oil tankers and weathers conditions as well as updated *predictions* of these factors. While tugs begin to move according to the solutions planned by the GA, new patrol trajectories can be calculated and replace the old ones. This strategy is equivalent to a RHC scheme, which is interchangeably termed model predictive control (MPC) in the literature (e.g., see [7,8] for theoretical treatments).

In RHC, a control strategy that minimises some cost function is calculated a prespecified duration, namely the prediction horizon, into the future. Only the first portion of this strategy is implemented before another control strategy is calculated based on new and predicted information available. The new solution replaces the old one but again only the first portion is implemented. This process repeats as a sequence of RHC steps.

RHC is currently one of the most popular control algorithms employed in computer-controlled systems, predominantly in the petrochemical industry, but also increasingly so in electromechanical control problems (e.g., see [9]). It can be shown that RHC can be designed with guaranteed asymptotic closed-loop stability [9] and this remarkable property is perhaps the most important reason for its popularity.

Constraints. An advantage of using RHC is that constraints can be handled in the design phase and not post hoc (e.g., see [9,7]). For tugs, such a constraint is the inherent limitation of moving no faster than their maximum speed. This speed limits the size of the envelopes in Fig. 2 and thus the number of reachable cross points. Using RHC it is possible to incorporate this constraint in the planning of tug trajectories.

Optimisation. A good choice of initial population allows the GA to find good solutions in fewer iterations than simply using a random population. It is possible to take the dynamics of the simulated scenario into account and, assuming that the scenario will not change significantly, a solution found at one RHC step should also be a viable solution at the next RHC step. This is achieved by an elitist strategy of keeping (a slightly modified version of) the best chromosome at one RHC step and inserting it into the initial population of the GA at the next RHC step.

2.3 Simulation Study

The technical computing software package MATLAB¹ was used for the implementation of a simulation study of the tug positioning problem. A number of choices had to be made about properties of oil tankers and patrol tugs, the GA and RHC, and general settings. Based on preliminary work [10] and extended testing, the settings described below were chosen.

¹ MATLAB R2010b, available at www.mathworks.com

Number of Ships. Based on information provided by NCA staff or affiliates and a recent report [1], it was decided to use $N_p = 3$ tugs and $N_o = 6$ oil tankers for the simulations. Whereas these numbers were realistic as of 2010, they will increase significantly the next decades due to the development of oil and gas fields in the area (see Sect. 4.5).

Position of Ships. The initial position of oil tankers at time $t = 0$ was varied for each simulation, with oil tankers being placed on z (in km) at positions drawn randomly from a uniform distribution in a 1500-km range from $z = -750$ to $z = 750$. Dividing the same range on y into $N_p = 3$ equally-sized segments of length 500 km, the patrol tugs were always positioned initially at tug bases located in the centre of these segments, namely at $y = -500, 0, 500$. The reason for this was to compare the performance of the actively patrolling tugs controlled by the RHGA with keeping the patrol tugs on stand-by at uniformly distributed stationary bases.

Velocities of Ships. According to [1], oil tankers have a typical operating speed of 14–15 knots whereas tugs have a global average maximum speed of about 12 knots, spanning from 5–26 knots [2]. In the geographical area of this case study, the typical maximum speed of tugs is 15 knots and operating speed of oil tankers is 10–14 knots [3].

Based on these figures, each oil tanker was initialised with a random speed in either the negative (southbound) or positive (northbound) y -direction and drawn from a uniform distribution in the range $\pm[20, 30]$ (km/h). The oil tankers maintained their respective speeds throughout each simulation.

The patrol tugs were assigned a maximum speed of 30 km/h, corresponding to the envelopes presented previously in Figure 2.

Drift Trajectories. Wind, wave heights, ocean currents, oil tanker size and shape, and other factors lead to nonlinear drift trajectories perhaps resembling those in Fig. 1. To implement nonlinearity, it was assumed that any oil tanker in drift will follow an eastbound sinusoidal trajectory with period equal to T_h scaled by its velocity v [3]. That is, if the c th oil tanker with velocity v and position $z(t_d)$ loses manoeuvrability at $t = t_d$, it is predicted to drift across the patrol line at

$$y_{t_d+\hat{\Delta}}^c = z(t_d) + v \sin\left(\frac{2\pi}{T_h}\hat{\Delta}\right) \quad (8)$$

after a predicted drift time $\hat{\Delta}$.

For each oil tanker, a random integer drawn from a uniform distribution $[8, 9 \dots, 12]$ was chosen as its predicted drift time and kept constant throughout each simulation. According to [2], drift times of only 10 hours are considered fast drift, whereas slow drift means that most tankers will not run aground for the first 20–30 hours of uncontrolled drift. Thus, the choice of drift times in the interval 8–12 hours is a conservative estimate. In most cases, tugs will have more time to come to the rescue of a drifting ship.

² Information provided by a close affiliate of the NCA.

³ Note that this relationship is not physically realistic but simply chosen for the sake of introducing nonlinearity.

GA Settings. At every RHC step, the GA was set to perform $N_{\text{iter}} = 100$ iterations searching for a solution set of optimal patrol trajectories minimising the cost function given by (3). As discussed in Sect. 4.3 each RHC step keeps a modified version of the best chromosome found in the previous RHC step. This ensures that much fewer iterations are needed in later RHC steps than early ones.

The population size was set to 10 chromosomes, the mutation rate was set to 0.1, and the selection parameter was set to 0.5. Together with the other simulation parameters, these choices gave a good tradeoff between exploration and exploitation.

RHC Settings. The GA was used to search for optimal trajectories with a duration of $T_h = 24$ hours for the patrol tugs. At every RHC step, each of duration $T_s = 1$ hour, only the first sample of these trajectories was executed by each tug before another solution set of trajectories was generated by the GA. This process was repeated for $N_{\text{RHC}} = 26$ RHC steps, yielding scenarios simulated from $t_d = 0$ to $t_d = 25$ hours.

General Settings. A total of $N_{\text{sim}} = 30$ scenarios (random initial positions, velocities, and drift times of oil tankers) were simulated. For each scenario i , the minimum costs found by the GA at each RHC step were calculated and the average cost stored as the i th element in a vector \mathbf{f}_{RHGA} of length N_{sim} . Similarly, the costs incurred if the patrol tugs stayed on stand-by at their individual bases were calculated and the average stored as the i th element in a vector $\mathbf{f}_{\text{static}}$ of length N_{sim} .

Settings Summary. The simulation settings are summarised in Table 1.

3 Results

3.1 Simulation Example

Figure 3 shows a simulation example using the settings given in Table 1. Initially at time $t_d = 0$ (Fig. 3(a)), three patrol trajectories, each of duration $T_h = 24$ hours, are planned for the tugs based on the predicted distributions of cross points. The first tug at $y_0^1 = -500$ is assigned the task of covering the isolated bottom cluster of cross points centred around $y = -600$, whereas the second tug at $y_0^2 = 0$ is assigned the top cluster of cross points centred around $y = 100$. The third tug at $y_0^3 = 500$ is not assigned any cross points and given a don't care, or random walk, trajectory.

In Fig. 3(b), the positions of oil tankers and patrol tugs are shown for $t_d = 5$. Because of the last five hours of oil tanker movements, there are now three distinct clusters of cross points. The GA now performs task reassignment by planning for the top tug to cover the top cluster of cross points. The middle tug is assigned the middle cluster, and the bottom tug the bottom cluster.

The remaining Figs. 3(c)-3(f) shows how the scenario develops for $t_d = 10, 15, 20, 25$, with the three tugs constantly being assigned and tracking sets of cross points, whose positional distributions change with time.

In terms of performance one may compare the cost of the RHGA-generated trajectories to that of static trajectories, that is, keeping each patrol tug stationary at its base. For this simulation example, which was scenario number 4, the static cost was 10844, whereas the RHGA cost was 3130, representing a cost reduction of 71.1 %.

Table 1. Simulation settings

Oil tanker settings	
Number of tankers N_o	6
Random initial position (km)	$[-750, 750]$
Random velocity (km/h)	$\pm[20, 30]$
Drift field	sinusoidal eastbound
Random drift time $\hat{\Delta}$ (hours)	$[8, 9, \dots, 12]$
Patrol tug settings	
Number of tugs N_p	3
Initial positions (km)	$\{-500, 0, 500\}$
Max velocity (km/h)	± 30
GA settings	
Iterations N_{iter}	100
Population size	10
Mutation rate	0.1
Selection	0.5
RHC settings	
Prediction horizon T_h (hours)	24
Simulation step size T_s (hours)	1
Number of steps N_{RHC}	26
General settings	
Number of scenarios N_{sim}	30
Strategies	RHGA, static

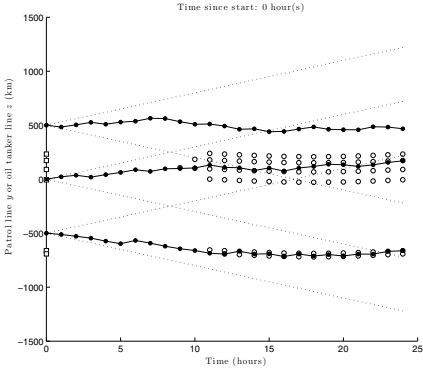
3.2 Main Study

Table 2 summarises the results from simulation scenario number 4 in Fig. 3 and 29 other simulated scenarios based on the settings presented in Table 1. For every scenario, the costs of each RHC step were summed and averaged for both the static case and the RHGA case. The mean cost for the 30 scenarios was 7372 for the RHGA and 17342 for the static strategy. This represents a mean improvement, or performance, of 57.5 % by the RHGA.

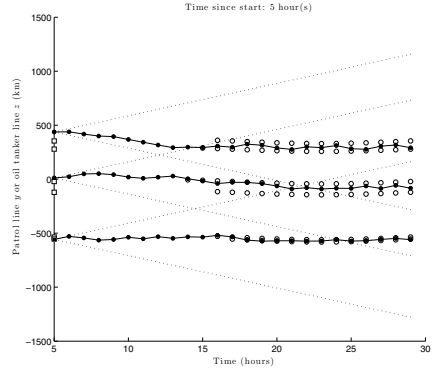
Comparing the standard deviation (STD) of the costs of the static case and the RHGA case, the STD of the RHGA was smaller by 34.6 %. However, because the static case has a much higher mean, its relative STD of 0.211 is 53.9 % smaller than the relative STD of the RHGA. Hence, relative to respective means, the cost of the static strategy varied less (it is consistently high) than that of the RHGA.

The minimum cost for a single scenario was 10844 for the static strategy and 3130 for the RHGA. Incidentally, the minimum cost occurred in scenario 4 for both cases. Inspection⁴ of this simulated scenario showed that cross points were distributed close to tug bases throughout the simulation, thus the static strategy resulted in a low cost. Still, the RHGA outperformed the static solution by 71.1 %.

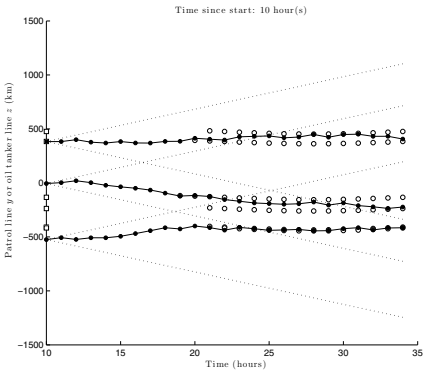
⁴ Where no figure is referred to the reader must trust the inspection made by the author.



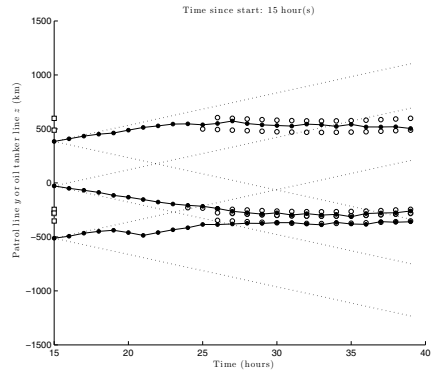
(a) $t_d = 0$



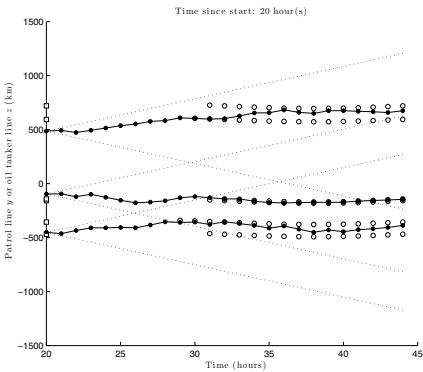
(b) $t_d = 5$



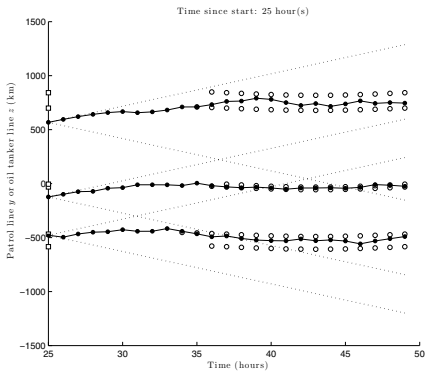
(c) $t_d = 10$



(d) $t_d = 15$



(e) $t_d = 20$



(f) $t_d = 25$

Fig. 3. Example simulation

The maximum cost for a single scenario for the static case was 22846 (scenario 2). Inspection of this scenario revealed that for most of the RHC steps, a large portion of cross points was distributed far to the south of the nearest southernmost tug base and

Table 2. Simulation results

Statistic	f_{static}		f_{RHGA}		Reduction by RHGA	
Mean	17342		7372		57.5 %	
STD	3667		2399		34.6 %	
Relative STD	0.211		0.325		-53.9 %	
Minimum (scenario)	10844	(4)	3130	(4)	31.0 %	(3)
Maximum (scenario)	22846	(2)	12503	(23)	79.0 %	(2)
95 % bounds	11309	or more	11318	or less	-0.1 %	

thus the static strategy had a high cost for this scenario. The RHGA solution, on the other hand, was very good at this scenario with a cost of only 4796, thus outperforming the static solution by 79.0 %, which was also the highest cost reduction by the RHGA for any of the scenarios.

The maximum cost for a single scenario for the RHGA was 12503 (scenario 23). Inspection of this scenario showed that a small cluster of cross points was located far away to the south from the other cross points. Minimisation of the cost function by the RHGA resulted in a solution where patrol trajectories ignored this small cluster, which contributed to the high cost. The RHGA still outperformed the static case by 37.6 %.

The worst performance in terms of cost reduction by the RHGA occurred in scenario 3, where the static solution had a cost of 16544 and the RHGA had a cost of 11415, or a reduction of only 31.0 %. Inspection of this scenario showed that cross points were divided into six separate clusters, each far away from the others. The static solution performed better than its average cost for this scenario, which is unsurprising, given that uniformly spread out tug bases is a good choice for uniformly spread out cross points.

Finally, subtracting $1.645 \times \text{STD}$ from the mean cost of the static strategy shows that an estimated 95 % of all scenarios in the static case will have a cost *greater* than approximately 11300. Similarly, adding $1.645 \times \text{STD}$ to the mean cost of the RHGA shows that an estimated 95 % of all simulated scenarios employing the RHGA will have a cost *smaller* than approximately 11300. The likelihood that the static solution is at least as good as the RHGA solution is thus very small.

3.3 Conclusions

The simulation results show that the RHGA is able to simultaneously perform coordinated control, task assignment, and multiple target tracking in a dynamic environment. Based on current and predicted information, a GA calculates patrol trajectories that minimise a cost function. However, as the environment changes, an RHC process must be employed, where the GA constantly replans new trajectories based on the most recent data.

Employing a cost function related to the distance from each cross point to the nearest predicted patrol trajectory gives good tracking but also provides task assignment for free. The resulting patrol trajectories suggested by the RHGA yield good prevention against possible drift accidents due to taking the predicted future environment into account.

4 Discussion

The simulation study presented here is substantially updated and extended compared to a preliminary study presented previously [10]. Modifications that have been made include realistic movement distances and speeds of tugs and oil tankers, nonlinear drift and cross point trajectories, an improved static strategy with tugs on stand-by uniformly positioned at stationary bases, a more rigorous problem formulation and definition of cost function, and a detailed analysis of the results.

In the following, some important aspects of the present study will be discussed.

4.1 Evaluation of Performance

Performance was measured by comparing the RHGA with a very simple static strategy where tugs are kept stationary on stand-by at bases located uniformly along the coast-line. The static method is good for very large and uniform distributions of cross points. For small numbers of oil tankers, on the other hand, this method does not perform well because cross points will often exist far from tug bases. As demonstrated in Sect. 3 the RHGA significantly outperformed the static strategy for all simulated scenarios.

An alternative to the static method is a simple heuristic method such as letting patrol trajectories move towards the nearest cross point. Preliminary studies not presented here show that this method performs well when the numbers of tugs and tankers are approximately equal but as the number of tankers increases its performance drops significantly compared to that of the RHGA. The reason, of course, is that when all tugs have been allocated a distributions of cross points, superfluous distributions will be ignored, which in turn causes evaluations of the cost function to increase drastically.

It still remains to compare the RHGA with other intelligent algorithms for the same problem as defined in this paper. Nevertheless, the results from this study are very promising and shows that the RHGA provides a viable method for solving dynamic resource allocation problems of the kind presented here.

4.2 Choice of Cost Function

Choosing a suitable cost function is essential for a GA to be able to solve the problem at hand. Although the selected cost function (3) seems to be a reasonable choice, there are likely other choices that may be equally, or better, suited to our problem.

A potential modification to the cost function is to include a term for the control input in order to punish excessive fuel consumption. If so, care must be taken to ensure that this does not compromise the main goal of covering cross points and reducing the overall risk picture.

Another option is to introduce risk weights on oil tankers and scale the minimum distances in the cost function by these weights. Such risk weights already exists in the models of the NCA.

Finally, it would be interesting to let tugs have different maximum speeds and also let the speeds of tugs and oil tankers vary over time due to weather conditions, cargo and fuel effects, and other factors. In this case, the cost function would have to be modified to sum minimum *rescue times*, and not minimum distances.

4.3 Optimisation

For a slow-changing dynamic environment, a good chromosome at one time instant is likely a good chromosome at the next. Consequently, as described in Sect. 2.2, the RHGA keeps the best chromosome from one RHC step and places it in the initial population of the next. The other chromosomes are randomly initialised as usual.

If desirable, this strategy can be used to reduce the overall number of GA iterations since only a fraction of the initial number of iterations is needed for subsequent RHC steps. This is because the dynamics are slow-varying and the GA will tune in to good solution spaces where previously found solutions greatly assists the GA in finding new, good solutions.

4.4 Real-Time Requirements

Simulating a single RHC step with three tugs and six oil tankers for a particular scenario took about 30 seconds on a MacBook Pro Core 2 Duo 2.53 GHz computer. Increasing the number of oil tankers tenfold to 60 (which might be realistic in the not too distant future, see Section 4.5), one RHC step took slightly less than five minutes. This shows that the RHGA can accommodate much greater complexity than simulated in this study while staying within the real-time requirement of finishing each RHC step within an hour of real-time. It also implies that more accurate solutions can be obtained by increasing GA parameters such as population size and number of iterations at each RHC step.

Conversely, the small execution time for a RHC step means that the simulated duration of a RHC step can be greatly reduced if desired. This may not be relevant for the study presented here but implies that systems with much faster dynamics may take advantage of the RHGA. An example where each RHC step must be in the range of tenths of seconds or smaller is real-time control of football-playing robots, where algorithm speed will most definitely be an issue. For such applications, it is possible to adjust the GA and RHC settings to obtain small RHC step durations as required. Specifically, one may reduce the prediction horizon, number of iterations, and population size. This may not necessarily degrade performance. For example, employing a large prediction horizon in a football game where it is only possible to predict actions a short period ahead will not increase performance, it may even degrade it if it causes each RHC step to take too long.

4.5 Other Simulation Scenarios

Based on recent information in a governmental report made by the Norwegian Institute of Maritime Research [1], the choice of three tugs and six oil tankers represents a realistic and typical scenario as of today. Nevertheless, due the development of large oil and gas fields in the Barents Sea such as Goliat, Snhvit, and Shtokman, oil and gas tanker traffic will drastically increase over the next 10–15 years. As mentioned in Section 4.1, the RHGA can easily handle the tenfold number of oil tankers while maintaining real-time requirements and thus appears well suited for much heavier traffic than that of today.

Cases where drifting actually occurs or situations where a tug becomes unavailable due to refuelling, change of crew, or being busy rescuing a drifting tanker have not been simulated. Moreover, no attempts have been made at trying to estimate how many tugs are necessary to maintain a sufficient degree of safety for a given number of tankers. These issues are highly relevant considering the foreseen increase in tanker activity.

Finally, it would be of interest to include more realistic two-dimensional (2D) planning for ships and three-dimensional (3D) planning for aeroplanes or submersible vehicles. This should be investigated further, particularly in light of other applications where dynamic resource allocation takes place at higher frequencies than for the tug positioning problem.

4.6 Other Applications

In addition to the one-dimensional (1D) problem described in this paper, the RHGA could be modified for performing multi-target allocation and tracking also in 2D and 3D dynamic environments. A 2D version in environments with slow dynamics may not require huge modifications, however, for fast dynamics and/or 3D environments, the algorithm must be improved, for example through distributed evaluation of the cost function.

Moreover, it could be interesting to combine the RHGA with so-called boid, or flocking, rules involving cohesion, separation, and alignment [12]. In a promising effort, [13] presents a flocking algorithm that modifies the flocking rules by [12] and succeeds in multi-target tracking performed by multiple agents. Through further development, a modified version of the RHGA could perform equally well as the algorithm used for the scenarios described by [13].

Furthermore, the problem definition used in this paper somewhat resembles that of the RoboFlag Drill described by [14]. They describe a scenario where a set of defenders are guarding a circular defence zone against a set of attackers. The attackers are randomly placed in an outer circle circumscribing the inner defence zone and move with constant velocity towards the zone. The goal of the defenders is to intercept as many of the incoming attacking trajectories before they reach the defence zone. It would be of great interest to test the RHGA for this scenario and compare the results with those of [14].

4.7 Concluding Remarks

This paper shows that a GA combined with RHC is able to simultaneously perform coordinated control, task assignment, and multiple target tracking in dynamically changing environments. The problem description is an interesting and non-trivial challenge for researchers in the field who are welcome to find alternative methods for solving it.

Acknowledgements. The author wishes to thank staff at the NCA and Christian Michelsen Research for providing details on oil tanker traffic and the dynamic positioning of tugs operating along the northern Norwegian coast. Collaboration with the NCA has recently been formalised and it is envisioned that a future version of the RHGA will be part of the decision support tools at VTS centres run by the NCA.

In addition, the author is grateful for the contributions of his colleagues, associate professor Siebe B. van Albada and professor Harald Yndestad at the Ålesund University College, in the early phase of this work.

References

1. Havforskningsinstituttet: Fisken og havet, særnummer 1a-2010: Det faglige grunnlaget for oppdateringen av forvaltningsplanen for Barentshavet og havområdene utenfor Lofoten. Technical report, Institute of Marine Research (Havforskningsinstituttet) (2010)
2. Eide, M.S., Endresen, Ø., Breivik, Ø., Brude, O.W., Ellingsen, I.H., Røang, K., Hauge, J., Brett, P.O.: Prevention of oil spill from shipping by modelling of dynamic risk. *Marine Pollution Bulletin* 54, 1619–1633 (2007)
3. Eide, M.S., Endresen, Ø., Brett, P.O., Ervik, J.L., Røang, K.: Intelligent ship traffic monitoring for oil spill prevention: Risk based decision support building on AIS. *Marine Pollution Bulletin* 54, 145–148 (2007)
4. Russell, S., Norvig, P.: *Artificial Intelligence: A Modern Approach*. Pearson (2010)
5. Haupt, R.L., Haupt, S.E.: *Practical Genetic Algorithms*, 2nd edn. Wiley (2004)
6. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional (1989)
7. Maciejowski, J.M.: *Predictive Control with Constraints*, 1st edn. Prentice Hall (2002)
8. Rossiter, J.A.: *Model-based Predictive Control*. CRC Press (2004)
9. Goodwin, G.C., Graebe, S.F., Salgado, M.E.: *Control System Design*. Prentice Hall, New Jersey (2001)
10. Bye, R.T., van Albada, S.B., Yndestad, H.: A receding horizon genetic algorithm for dynamic multi-target assignment and tracking: A case study on the optimal positioning of tug vessels along the northern norwegian coast. In: *Proceedings of the International Conference on Evolutionary Computation*, pp. 114–125. SciTePress (2010)
11. Det Norske Veritas: Rapport Nr. 2009-1016. Revisjon Nr. 01. Tiltaksanalyse — Fartsgrenser for skip som opererer i norske farvann. Technical report, Sjøfartsdirektoratet (2009)
12. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: *Computer Graphics (ACM SIGGRAPH)*, vol. 21, pp. 25–34 (1987)
13. Luo, X., Li, S., Guan, X.: Flocking algorithm with multi-target tracking for multi-agent systems. *Pattern Recognition Letters* 31, 800–805 (2010)
14. Earl, M.G., D’andrea, R.: A decomposition approach to multi-vehicle cooperative control. *Robotics and Autonomous Systems* 55, 276–291 (2007)

Part II
Fuzzy Computation

Generating Optimized Fuzzy Partitions to Classification and Considerations to Management Imprecise Data

J.M. Cadenas, M.C. Garrido, and R. Martínez

University of Murcia, Faculty of Informatic, Campus of Espinardo, 30100 Murcia, Spain
{jcadenas, carmengarrido, raquel.m.e}@um.es

Abstract. Many algorithms for classification need to discretize the continuous attributes for their development. Therefore the discretization of continuous attributes is a very important part in data mining. In this paper, we propose a technique for discretizing continuous attributes by means of a series of fuzzy sets which constitute a fuzzy partition of the domain of these attributes. The definition of these sets is very important as it affects the results obtained in the classification algorithms. Throughout this document we present a strategy to construct fuzzy sets in order to improve classification results. Additionally, we give some ideas about how to improve this strategy in order to work with another kinds of data. Also, we show various experimental results which evaluate our proposal in comparison with previously existing ones and where the results have been statistically validated.

Keywords: Fuzzy discretization, Fuzzy set, Genetic algorithm, Fuzzy decision tree.

1 Introduction

The selection, processing and data cleaning is one of the phases making up the process of knowledge discovery. This phase can be very important for some algorithms of classification, because the data must be preprocessed so that the algorithm can work with them. A possible change in the data may be the discretization of continuous values. The discretization continuous attributes can be carried out through crisp partitions or fuzzy partitions. Crisp partitions use classical logic, where each attribute is split into several intervals, whereas fuzzy partitions use fuzzy logic. On the one hand, we can find techniques to discretize continuous attributes into crisp intervals, [5], [12], in which a domain value can only belong to a partition or interval. On the other hand we find methods to discretize continuous attributes into fuzzy intervals [10], [11], in this case, a domain value can belong to more than one element of the fuzzy partition.

In this study we present the OFP_CLASS Algorithm to carry out a fuzzy discretization of continuous attributes, which is divided in two stages. In the first stage, we carry out a search of split points for each continuous attribute. In the second stage, based on these split points, we use a genetic algorithm which optimizes the fuzzy sets formed from the split points. Having designed the fuzzy sets that make up the fuzzy partition of each continuous attribute, they are evaluated with a classifier constituted by a fuzzy decision tree.

The structure of this study is as follows. In Section 2, we are going to present a taxonomy of discretization methods, as well a review various discretization methods. Then, in Section 3, we are going to present the OFP_CLASS Algorithm, which is based on a decision tree and a genetic algorithm, as our proposal for the problem of fuzzy discretization applied to classification. In Section 4, we propose an extension of the OFP_CLASS Algorithm to incorporate interval values. Next, in Section 5, we will show various experimental results which evaluate our proposal in comparison with previously existing ones and where the results have been statistically validated. Finally, in Section 6, we will show the conclusions of this study.

2 Discretization Methods

The classification is one of the most important topics in data mining area. There are many different classification methods which work with continuous values and/or discrete values. However, not all classification algorithms can work with continuous data, hence discretization techniques are needed to these algorithms. Although, there are algorithms which work with discrete data by the fact that they can improve results in classification tasks.

When a discretization process is to be developed, four iterative stages must be carried out, [12]:

1. The values in the database of the continuous attributes to be discretized are ordered.
2. The best split point for partitioning attribute domains in the case of top-down methods is found, or the best combination of adjacent partitions for bottom-up methods is found.
3. If the method is top-down, once the best split point is found, the domain of each attribute is divided into two partitions, and when the method is bottom-up, both partitions are merged.
4. Finally, we check whether the stopping criterion is fulfilled, and if so the process is terminated.

In this general discretization process we have differentiated between top-down and bottom-up algorithms. However, there are more complex taxonomies for the different methods of discretization such as that presented in [12] and which are shown here:

- Supervised or non-supervised. Non-supervised methods are those based solely on continuous attribute value in order to carry out discretization, whereas supervised ones use class value to discretize continuous attributes, so that they are more or less uniform with regard to class value.
- Static or dynamic. In both types of methods it is necessary to define a maximum number of intervals and they differ in that static methods seek to divide each attribute in partitions sequentially, whereas dynamic ones discretize domains by dividing all the attributes into intervals simultaneously.
- Local or global. Local methods of discretization are those which use algorithms such as C4.5 or its successor C5.0, [14], and they are only applied to specific regions in the database. On the other hand, global methods are based on the whole database to carry out discretization.

- Top-down or bottom-up. Top-down methods begin with an empty list of split points and add them as the discretization process finds intervals. On the other hand, bottom-up methods begin with a list full of split points and eliminate points during the discretization process.
- Direct or incremental. Direct methods divide the dataset directly into k intervals. Therefore they need an external input determined by the user to indicate the number of intervals. Incremental methods begin with a simple discretization and undergo an improvement process. For this reason they need a criterion to indicate when to stop discretizing.

In addition to the taxonomy exposed, from another viewpoint we consider discretization methods can also be classified according to the type of partitions constructed, crisp or fuzzy partitions.

Thus, in the literature we find some algorithms that generate crisp partitions. Among these, in [9] describes a method that performs crisp intervals taken as a measure the amplitude or frequency, which need to fix a k number of intervals. Also, [9] describes other method, called R1, which needs to have a fixed number of k intervals, but in this case, the measure which used is the class label. Another method that constructs crisp partitions, D2, is described in [5], where the measure used is entropy.

On the other hand, we find methods which discretize continuous values in fuzzy partitions, in this case, these methods use decision trees, clustering algorithms, genetic algorithms, etc. So, in [1] a hierarchical fuzzy partition based on $2^{|A|}$ -tree decomposition is carried out, where $|A|$ is the number of attributes in the system. This decomposition is controlled by the degree of certainty of the rules generated for each fuzzy subspace and the deeper hierarchical level allowed. The fuzzy partitions formed for each domain are symmetric and triangular. Furthermore, one of the most widely used algorithms for fuzzy clustering is fuzzy c-means (FCM) [4]. The algorithm assigns a set of examples, characterized by their respective attributes, to a set number of classes or clusters. Some methods developed for fuzzy partitioning start from the FCM algorithm and add some extension or heuristic to carry out an optimization in the partitions. We can find some examples in [10], [11]. Also, a method that constructs fuzzy partition using a genetic algorithm is proposed in [13], where fuzzy partitions are obtained through beta and triangular functions. The construction process of fuzzy partitions is divided into two stages. In the first stage, fuzzy partitions with beta [7] or triangular functions are constructed; and in the second stage these partitions are adjusted with a genetic algorithm.

3 OFP_CLASS: An Algorithm to Generate Optimized Fuzzy Partitions to Classification

In this section, the OFP_CLASS Algorithm we propose for discretizing continuous attributes by means of fuzzy partitions is presented and it is may be catalogued as supervised and local. The OFP_CLASS Algorithm is made up of two stage. In the first stage, crisp intervals are defined for each attribute. In the second stage, these intervals obtained are used to form an optimal fuzzy partition for classification using a genetic algorithm, but not all the crisp intervals obtained are used, because the genetic

algorithm is who determines which intervals are the best. The partition obtained for each attribute guarantees the:

- Completeness (no point in the domain is outside the fuzzy partition), and
- Strong fuzzy partition (it verifies that $\forall x \in \Omega_i, \sum_{f=1}^{F_i} \mu_{B_f}(x) = 1$, where B_1, \dots, B_{F_i} are the F_i fuzzy sets for the partition corresponding to the i continuous attribute with Ω_i domain)

The domain of each i continuous attribute is partitioned in trapezoidal fuzzy sets, B_1, B_2, \dots, B_{F_i} , so that:

$$\mu_{B_1}(x) = \begin{cases} 1 & b_{11} \leq x \leq b_{12} \\ \frac{(b_{13}-x)}{(b_{13}-b_{12})} & b_{12} \leq x \leq b_{13} \\ 0 & b_{13} \leq x \end{cases} ; \quad \mu_{B_2}(x) = \begin{cases} 0 & x \leq b_{12} \\ \frac{(x-b_{12})}{(b_{13}-b_{12})} & b_{12} \leq x \leq b_{13} \\ 1 & b_{13} \leq x \leq b_{23} \\ \frac{(b_{24}-x)}{(b_{24}-b_{23})} & b_{23} \leq x \leq b_{24} \\ 0 & b_{24} \leq x \end{cases} ;$$

$$\dots ; \quad \mu_{B_{F_i}}(x) = \begin{cases} 0 & x \leq b_{(F_i-1)3} \\ \frac{(x-b_{(F_i-1)3})}{(b_{(F_i-1)4}-b_{(F_i-1)3})} & b_{(F_i-1)3} \leq x \leq b_{(F_i-1)4} \\ 1 & b_{F_i3} \leq x \end{cases}$$

Before going into a detailed description of OFP_CLASS Algorithm, we are going to introduce the nomenclature we are going to use throughout the section and then we will present the fuzzy decision tree to be used in the evaluation of the fuzzy partitions generated and which, with some modification, is used in the first stage of OFP_CLASS Algorithm.

3.1 Nomenclature and Basic Expressions

- N : Node which is being explored at any given moment.
- C : Set of classes or possible values of the decision attribute. $|C|$ denotes the C set cardinal.
- E : Set of examples from the dataset. $|E|$ denotes the number of examples from the dataset.
- e_j : j -th example from the dataset.
- A : Set of attributes which describe an example from the dataset. $|A|$ denotes the number of attributes that describe an example.
- G_i^N : information gain when node N is divided by attribute i .

$$G_i^N = I^N - I^{S_{V_i}^N} \quad (1)$$

where:

- I^N : Standard information associated with node N . This information is calculated as follows:
 1. For each class $k = 1, \dots, |C|$, the value P_k^N , which is the number of examples in node N belonging to class k is calculated:

$$P_k^N = \sum_{j=1}^{|E|} \chi_N(e_j) \cdot \mu_k(e_j) \tag{2}$$

where:

- $\chi_N(e_j)$ the degree of belonging of example e_j to node N .
 - $\mu_k(e_j)$ is the degree of belonging of example e_j to class k .
2. P^N , which is the total number of examples in node N , is calculated.

$$P^N = \sum_{k=1}^{|C|} P_k^N$$

3. Standard information is calculated as:

$$I^N = - \sum_{k=1}^{|C|} \frac{P_k^N}{P^N} \cdot \log \frac{P_k^N}{P^N}$$

- $I^{S_{V_i}^N}$ is the product of three factors and represents standard information obtained by dividing node N using attribute i adjusted to the existence of missing values in this attribute.

$$I^{S_{V_i}^N} = I_1^{S_{V_i}^N} \cdot I_2^{S_{V_i}^N} \cdot I_3^{S_{V_i}^N}$$

where:

- * $I_1^{S_{V_i}^N} = 1 - \frac{P^{N_{m_i}}}{P^N}$, where $P^{N_{m_i}}$ is the weight of the examples in node N with missing value in attribute i .
- * $I_2^{S_{V_i}^N} = \frac{1}{\sum_{h=1}^{H_i} P^{N_h}}$, H_i being the number of descendants associated with node N when we divide this node by attribute i and P^{N_h} the weight of the examples associated with each one of the descendants.
- * $I_3^{S_{V_i}^N} = \sum_{h=1}^{H_i} P^{N_h} \cdot I^{N_h}$, I^{N_h} being the standard information of each descendant h of node N .

3.2 A Fuzzy Decision Tree

In this section, we describe the fuzzy decision tree that we will use as a classifier to evaluate fuzzy partitions generated and whose basic algorithm will be modified for the first stage of the OFP_CLASS Algorithm, as we will see later.

The set of examples E out of which the tree is constructed is made up of examples described by attributes which may be nominal, discrete and continuous, and where there will be at least one nominal or discrete attribute which will act as a class attribute. The algorithm by means of which we construct the fuzzy decision tree is based on the ID3 algorithm, where all the continuous attributes have been discretized by means of a series of fuzzy sets.

An initial value equal to 1 ($\chi_{root}(e_j) = 1$) is assigned to each example e_j used in the tree learning, indicating that initially the example is only in the root node of the tree. This value will continue to be 1 as long as the example e_j does not belong to more than one node during the tree construction process. In a classical tree, an example can only belong to one node at each moment, so its initial value (if it exists) is not modified throughout the construction process. In the case of a fuzzy tree, this value is modified in two situations:

- When the example e_j has a missing value in an attribute i which is used as a test in a node N . In this case, the example descends to each child node N_h , $h = 1, \dots, H_i$ with a modified value as $\chi_{N_h}(e_j) = \chi_N(e_j) \cdot \frac{1}{H_i}$.
- According to e_j 's degree of belonging to different fuzzy partition sets when the test of a node N is based on attribute i which is continuous. In this case, the example descends to those child nodes to which the example belongs with a degree greater than 0 ($\mu_{B_f}(e_j) > 0$; $f = 1, \dots, F_i$). Because of the characteristics of the partitions we use, the example may descend to two child nodes at most. In this case, $\chi_{N_h}(e_j) = \chi_N(e_j) \cdot \mu_{B_f}(e_j)$; $\forall f | \mu_{B_f}(e_j) > 0$; $h = f$.

We can say that the $\chi_N(e_j)$ value indicates the degree with which the example fulfills the conditions that lead to node N on the tree.

The stopping condition is defined by the first condition reached out of the following:

(a) pure node, (b) there aren't any more attributes to select, (c) reaching the minimum number of examples allowed in a node. Having constructed the fuzzy tree, we use it to infer an unknown class of a new example:

Given the example e to be classified with the initial value $\chi_{root}(e) = 1$, go through the tree from the root node. After obtain the leaf set reached by e . For each leaf reached by e , calculate the support for each class. The support for a class on a given leaf N is obtained according to the equation (2). Finally, obtain the tree's decision, c , from the information provided by the leaf set reached and the value χ with which example e activates each one of the leaves reached.

In the following sections we describe the stages which comprise the Algorithm of discretization OFF_CLASS.

3.3 First Stage: Looking for Crisp Intervals

In this stage, a fuzzy decision tree is constructed whose basic process is that described in subsection 3.2, except that now a procedure based on priority tails is added and there are continuous attributes that have not been discretized. The discretization of these attributes is precisely the aim of this first stage.

To deal with non-discretized continuous attributes, the algorithm follows the basic process in C4.5. The thresholds selected in each node of the tree for these attributes

will be the split points that delimit the intervals. Thus, the algorithm that constitutes this first stage is based on a fuzzy decision tree that allows nominal attributes, continuous attributes discretized by means of a fuzzy partition, non-discretized continuous attributes, and furthermore it allows the existence of missing values in all of them. Algorithm 1 describes the whole process.

Algorithm 1. Search of crisp intervals

SearchCrispIntervals(*in* : E , *Fuzzy Partition*; *out* : *Split points*)

begin

1. Start at the root node, which is placed in the initially empty priority tail. Initially, the root node is found in the set of examples E with an initial weight of 1. The tail is a priority tail, ordered from higher to lower according to the total weight of the examples of nodes that form the tail. Thus the domain is guaranteed to partition according to the most relevant attributes.
2. Extract the first node from the priority tail.
3. Select the best attribute to divide this node using information gain expressed in equation (1) as the criterion. We can find two cases. The first case is where the attribute with the highest information gain is already discretized, either because it is nominal, or else because it had already been discretized earlier by the *Fuzzy Partition*. The second case arises when the attribute is continuous and non-discretized, in which case it is necessary to obtain the corresponding split points.
 - (a) If the attribute is already discretized, node N is expanded into as many children as possible values the selected attribute may have. In this case, the tree's behaviour is similar to that described in the Subsection 3.2.
 - (b) If the continuous attribute is not previously discretized, its possible descendants are obtained. To do this, as in C4.5, the examples are ordered according to the value of the attribute in question and the intermediate value between the value of the attribute for example e_j and for example e_{j+1} is obtained. The value obtained will be that which provides two descendants for the node and to which the criterion of information gain is applied. This is repeated for each pair of consecutive values of the attribute, searching for the value that yields the greatest information gain. The value that yields the greatest information gain will be the one used to divide the node and will be considered as a split point for the discretization of this attribute.
4. Having selected the attribute to expand node N , all the descendants generated are introduced in the tail according to the established order.
5. Go back to step two to continue constructing the tree until there are not nodes in the priority tail or until another stopping condition occurs, such as reaching nodes with a minimum number of examples allowed by the algorithm.

end

3.4 Second Stage: Constructing and Optimizing Fuzzy Partitions

Genetic algorithms are very powerful and very robust, as in most cases they can successfully deal with an infinity of problems from very diverse areas. These algorithms are normally used in problems without specialized techniques or even in those problems

where a technique does exist, but is combined with a genetic algorithm to obtain hybrid algorithms that improve results [6].

In this second stage of the OFF_CLASS Algorithm, we are going to use a genetic algorithm to obtain the fuzzy sets that make up the partitioning of continuous attributes of the problem. Given the $F_i - 1$ split points of attribute i obtained in the prior stage, we can define a maximum of F_i fuzzy sets that perform up the partition of i . The definition of the different elements that make up this genetic algorithm is as follows:

- **Encoding.** An individual will consist of two array v_1 and v_2 . The array v_1 has a real coding and its size will be the sum of the number of split points that the fuzzy tree will have provided for each attribute in the first stage.

Each gene in array v_1 represents the quantity to be added to and subtracted from each attribute's split point to form the partition fuzzy. On the other hand, the array v_2 has a binary coding and its size is the same that the array v_1 .

Each gene in array v_2 indicates whether the corresponding gene or split point of v_1 is active or not. The array v_2 will change the domain of each gene in array v_1 . The domain of each gene in array v_1 is an interval defined by $[0, \min(\frac{p_r - p_{r-1}}{2}, \frac{p_{r+1} - p_r}{2})]$ where p_r is the r -th split point of attribute i represented by this gene except in the first (p_1) and last (p_u) split point of each attribute whose domains are, respectively: $[0, \min(p_1, \frac{p_2 - p_1}{2})]$ and $[0, \min(\frac{p_u - p_{u-1}}{2}, 1 - p_u)]$.

When $F_i = 2$, the domain of the single split point is defined by $[0, \min(p_1, 1 - p_1)]$. The population size will be 100 individuals.

- **Initialization.** First the array v_2 in each individual is randomly initialized, provided that the genes of the array are not all zero value, since all the split points would be deactivated and attributes would not be discretized. Once initialized the array v_2 , the domain of each gene in array v_1 is calculated, considering what points are active and which not. After calculating the domain of each gene of the array v_1 , each gene is randomly initialized generating a value within its domain.
- **Fitness Function.** The fitness function of each individual is defined according to the information gain defined in [3]. Algorithm 2 implements the fitness function, where:
 - μ_{if} is the belonging function corresponding to fuzzy set f of attribute i .
 - E_k is the subset of examples of E belonging to class k .

This fitness function, based on the information gain, indicates how dependent the attributes are with regard to class, i.e., how discriminatory each attribute's partitions are. If the fitness we obtain for each individual is close to zero, it indicates that the attributes are totally independent of the classes, which means that the fuzzy sets obtained do not discriminate classes. On the other hand, as the fitness value moves further away from zero, it indicates that the partitions obtained are more than acceptable and may discriminate classes with good accuracy.

- **Selection.** Individual selection is by means of tournament, taking subsets with size 2.

Algorithm 2. Fitness Function**Fitness**(*in* : *E*, *out* : *ValueFitness*)**begin**1. For each attribute $i = 1, \dots, |A|$:1.1 For each set $f = 1, \dots, F_i$ of attribute i For each class $k = 1, \dots, |C|$ calculate the probability

$$P_{ifk} = \frac{\sum_{e \in E_k} \mu_{if}(e)}{\sum_{e \in E} \mu_{if}(e)}$$

1.2 For each class $k = 1, \dots, |C|$ calculate the probability

$$P_{ik} = \sum_{f=1}^{F_i} P_{ifk}$$

1.3 For each $f = 1, \dots, F_i$ calculate the probability

$$P_{if} = \sum_{k=1}^{|C|} P_{ifk}$$

1.4 For each $f = 1, \dots, F_i$ calculate the information gain of attribute i and set f

$$I_{if} = \sum_{k=1}^{|C|} P_{ifk} \cdot \log_2 \frac{P_{ifk}}{P_{ik} \cdot P_{if}}$$

1.5 For each $f = 1, \dots, F_i$ calculate the entropy

$$H_{if} = -\sum_{k=1}^{|C|} P_{ifk} \cdot \log_2 P_{ifk}$$

1.6 Calculate the I and H total of attribute i

$$I_i = \sum_{f=1}^{F_i} I_{if} \quad \text{and} \quad H_i = \sum_{f=1}^{F_i} H_{if}$$

2. Calculate the fitness as :

$$ValueFitness = \frac{\sum_{i=1}^{|A|} I_i}{\sum_{i=1}^{|A|} H_i}$$

end

- **Crossing.** The crossing operator is applied with a probability of 0.3, crossing two individuals through a single point, which may be any one of the positions on the vector. Not all crossings are valid, since one of the restrictions imposed on an individual is that the array v_2 should not have all its genes to zero. When crossing two individuals and this situation occurs, the crossing is invalid, and individuals remain in the population without interbreeding. If instead the crossing is valid, the domain for each gene of array v_1 is updated in individuals generated.
- **Mutation.** Mutation is carried out according to a certain probability at interval $[0.01, 0.1]$, changing the value of one gene to any other in the possible domain. First, the gene of the array v_2 is mutated and then checked that there are still genes with value 1 in v_2 . In this case, the gene in array v_2 is mutated and, in addition, the

domains of this one and its adjacent genes are updated in the vector v_1 . Finally, the mutation in this same gene is carried out in the vector v_1 .

If when a gene is mutated in v_2 all genes are zero, then the mutation process is not produced.

- **Stopping.** The stopping condition is determined by the number of generations situated at interval [150, 200].

The genetic algorithm should find the best possible solution in order to achieve a more efficient classification. By way of an example, let us suppose that we have a dataset that only consists of three attributes, for which the fuzzy decision tree has indicated 2, 3 and 1 split points for each one respectively and which we show in Table 1.

Table 1. Stage 1 of the OFP_CLASS Algorithm

Attribute 1	0.3	0.5	
Attribute 2	0.1	0.4	0.8
Attribute 3	0.7		

Based on the split points, in the second stage, the genetic algorithm will determine which of them will form the fuzzy partition of each attribute. Following the example, the domains of two possible individuals are showed in the Figure 1, where for each individual, the v_2 array is showed and the array v_1 shows the domain of the genes in which the corresponding gene in the array v_2 is 1. As we have already commented previously, the vector v_1 is made up of a set of values that represent for each attribute and split point what the distance to be added and subtracted to define the straight lines that make up the fuzzy sets. Also, the domain of each gene depends on previous and later active split points.

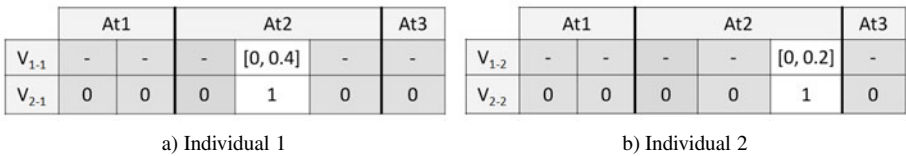


Fig. 1. Domains for each gene

Following with the example given, Figure 2 shows a possible valid crossing at point A. If the crossing is realized at point B instead of A, it would not be valid because the array v_2 would stay with all zero values and all the split points would be deactivated and attributes would not be discretized.

The mutation can generate invalid individuals too. The Figure 3 shows an example of mutation invalid, because the individual 2, after the crossing, only has one active

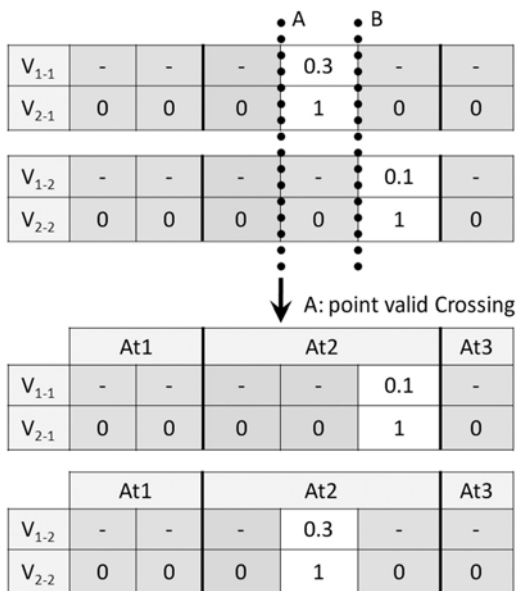


Fig. 2. Crossing example allowed

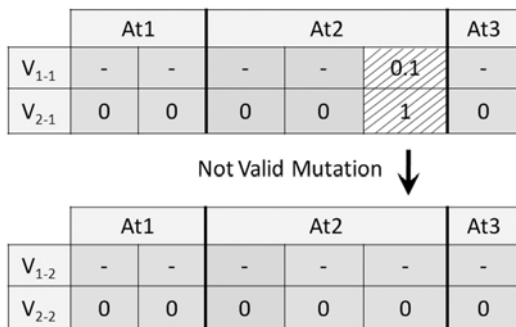


Fig. 3. Mutation example doesn't allowed

gene and whether this gene is turned off all genes in array v_2 are zero. If the mutated gene had been any other, the mutation would be valid. An important aspect is that if an inactive gene is mutated, then there are to calculate the domains of the mutated gene and adjacent.

If we assume, in the example, that individuals are not changed after the crossing shown in Figure 2 and the algorithm reaches a stopping condition, the algorithm only has discretized the second attribute in the two individuals. Figure 4 shows the discretization that each individual makes the second attribute.

This example shows that although in the first stage many split points are obtained, the algorithm may only use a subset of these to discretize.

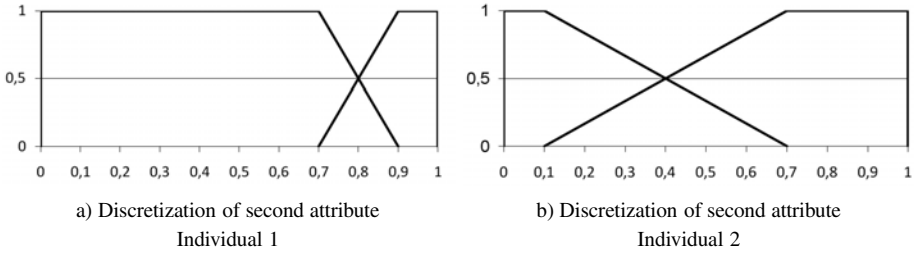


Fig. 4. Fuzzy Partition of the example

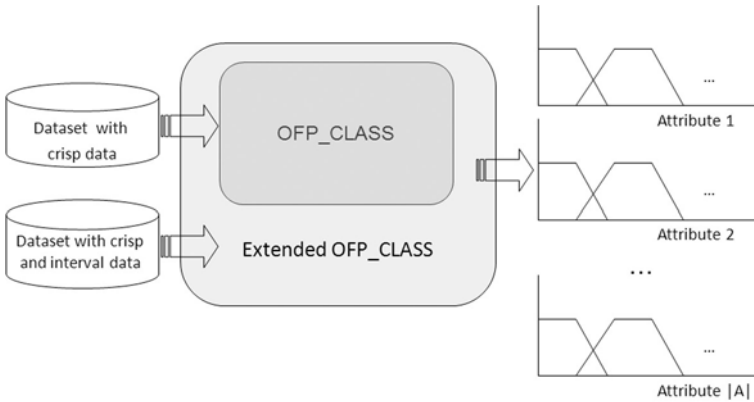


Fig. 5. Extending OFF_CLASS

4 Extending OFF_CLASS to Incorporate Interval Values

As we have shown through this work, OFF_CLASS obtains a fuzzy partition from a dataset available. The values of the different attributes of this dataset are known exactly, ie, they are crisp values.

However, imperfect information inevitably appears in domains and situations of real world so that the values of attributes may not be known exactly. Instrumental errors or corruption from noise during experiments may give rise to imprecise values when measuring a specific attribute. Therefore, we consider interesting to include the treatment of imprecise values in the OFF_CLASS Algorithm. So, in this section we propose a first extension of the OFF_CLASS Algorithm to obtain the fuzzy partition from attributes specified with crisp values and by intervals (classical imprecision). Figure 5 shows the idea of the extending process of OFF_CLASS Algorithm.

In order to carry out this extension, we need to modify the first stage of OFF_CLASS Algorithm.

- First we must modify the Algorithm 1 to allow the processing of interval values. The main modification should be performed in step 3 of that algorithm where the best attribute to divide a node using the information gain must be selected. In this

case to get the best split point for each attribute, is necessary to establish an order for the values in the dataset for each attribute.

This ordering of values must consider the existence of interval values and we will use some ordering indices with reasonable properties. A good analysis of the ordering indices can be seen in [16].

- Second, once selected the attribute and its threshold value to partition the node N , an example with interval value in that attribute descends to the first child node with a modified value $\chi_{N_h}(e_j)$ in proportion to the part of the interval that is below the threshold and to the second child node with a modified value $\chi_{N_h}(e_j)$ in proportion to the part of the interval that is above the threshold.

When the extension of OFP_CLASS to incorporate interval values is realized, it is straightforward to extend it to incorporate more general imprecision adding both fuzzy and interval values. Thus, the algorithm will work with imperfect data respecting its real nature and avoiding processing which may cause loss of information that can affect the performance of the algorithm and consequently the obtention of worse results.

5 Experiments

In this section we show several computational results which measure the accuracy of the OFP_CLASS Algorithm proposed. In order to evaluate the OFP_CLASS Algorithm a comparison with the results of [10] and [11] is carried out, in which fuzzy partitions are constructed by means of a combination of fuzzy clustering algorithms, using the majority vote rule or the weighted majority vote rule, respectively. To obtain these results we have used several datasets from the UCI repository [2], whose characteristics are shown in Table 2. It shows the number of examples ($|E|$), the number of attributes ($|A|$), the number of continuous attributes (Cont.) and the number of classes for each dataset (CL). “Abbr” indicates the abbreviation of the dataset used in the experiments.

In order to evaluate the partitions generated by the OFP_CLASS Algorithm, we classify the datasets using the fuzzy decision tree presented in Subsection 3.2. We compare the results obtained in [10] and [11] with those obtained by OFP_CLASS Algorithm. The comparison is carried out on the same datasets used in those two references. For this experiment, a 3×5 -fold cross validation was carried out. In Table 3, the best average

Table 2. Datasets Description

Dataset	Abbr	$ E $	$ A $	Cont.	CL
Australian Credit Approval	AUS	690	14	6	2
German Credit Data	GER	1000	24	24	2
Iris Data Set	IRP	150	4	4	3
Pima Indian Diabetes Data Set	PIM	768	8	8	2
SPECTF Heart Data Set	SPE	267	44	44	2
Thyroid Disease Data Set	THY	215	5	5	3
Zoo Data Set	ZOO	101	16	1	7

Table 3. Testing accuracies

Dataset	Best result of [10][11]	OFP_CLASS
AUS	60.29%	85.50%±0.00
GER	66.80%	73.13%±0.21
IRP	92.00%	97.33%±0.00
PIM	65.10%	77.07%±0.12
SPE	64.79%	84.09%±0.18
THY	79.07%	95.83%±0.00
ZOO	68.32%	94.06%±0.00

success percentages obtained in [10] and [11] and those obtained with OFP_CLASS Algorithm are shown. Also, in the case of OFP_CLASS Algorithm the standard deviation for each dataset is shown.

After the experimental results have been shown, we perform an analysis of them using statistical techniques. Following the methodology of [8] we use nonparametric tests. We use the Wilcoxon signed-rank test to compare two methods. This test is a non-parametric statistical procedure for performing pairwise comparison between two methods. Under the null-hypothesis, it states that the methods are equivalent, so a rejection of this hypothesis implies the existence of differences in the performance of all the methods studied. In order to carry out the statistical analysis we have used the tool R, [15].

Results obtained on comparing the OFP_CLASS Algorithm with the best result of [10] and [11] for each dataset show that, with a 99.9% confidence level, there are significant differences between the methods, with the OFP_CLASS Algorithm being the best.

6 Conclusions

In this study we have presented an algorithm for fuzzy discretization of continuous attributes, which we have called OFP_CLASS Algorithm. The aim of this algorithm is to find a partition that allows good results to be obtained when using it afterwards with fuzzy classification techniques. The algorithm makes use of two techniques: a Fuzzy Decision Tree and a Genetic Algorithm. Thus the proposed algorithm consists of two stages, using in the first of them the fuzzy decision tree to find divisions in the continuous attribute domain, and in the second, the genetic algorithm to find, on the basis of prior divisions, a fuzzy partition. In addition, we have discussed some ideas on how to make a fuzzy partition of the continuous attributes domains when the values may be given by interval values.

We have presented experimental results obtained by applying the OFP_CLASS Algorithm to various datasets. On comparing the results of the OFP_CLASS Algorithm with those obtained by two methods in the literature we conclude that the OFP_CLASS Algorithm is an effective algorithm and it obtains the best results. Moreover, all these conclusions have been validated by applying statistical techniques to analyze the behaviour of the algorithm.

Acknowledgements. Supported by the project TIN2008-06872-C04-03 of the MICINN of Spain and European Fund for Regional Development. Thanks also to the Funding Program for Research Groups of Excellence with code 04552/GERM/06 granted by the “Fundación Séneca”. R. Martínez is supported by the scholarship program FPI from the “Fundación Séneca” of Spain.

References

1. Ait Kbir, M., Maalmi, K., Benslimane, R., Benkirane, H.: Hierarchical fuzzy partition for pattern classification with fuzzy if-then rules. *Pattern Recognition Letters* 21(6-7), 503–509 (2000)
2. Asuncion, A., Newman, D.: Uci machine learning repository. University of California, School of Information and Computer Science, <http://www.ics.uci.edu/mllearn/MLRepository.html>
3. Au, W.H., Chan, K.C., Wong, A.: A fuzzy approach to partitioning continuous attributes for classification. *IEEE Tran. Knowledge and Data Engineering* 18(5), 715–719 (2006)
4. Bezdek, J.: *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell (1981)
5. Catlett, J.: N changing continuous attributes into ordered discrete attributes. In: *Fifth European Working Session on Learning*, pp. 164–177 (1991)
6. Cox, E.: *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*. Morgan Kaufmann Publishers, New York (2005)
7. Cox, E., Taber, R., O’Hagan, M.: *The Fuzzy Systems Handbook*. P. Professional, 2nd edn. (1998)
8. García, S., Fernández, A., Luengo, J., Herrera, F.: A study statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Computing* 13(10), 959–977 (2009)
9. Holte, R.: Very simple classification rules perform well on most commonly used datasets. *Machine Learning* 11, 63–90 (1993)
10. Li, C.: A combination scheme for fuzzy partitions based on fuzzy majority voting rule. In: *International Conference on Networks Security, Wireless Communications and Trusted Computing*, vol. 2, pp. 675–678 (2009)
11. Li, C., Wang, Y., Dai, H.: A combination scheme for fuzzy partitions based on fuzzy weighted majority voting rule. In: *International Conference on Digital Image Processing*, pp. 3–7 (2009)
12. Liu, H., Hussain, F., Tan, C., Dash, M.: Discretization: an enabling technique. *Journal of Data Mining and Knowledge Discovery* 6(4), 393–423 (2002)
13. Piero, P., Arco, L., García, M., Acevedo, L.: Algoritmos genéticos en la construcción de funciones de pertenencia borrosas. *Revista Iberoamericana de Inteligencia Artificial* 18, 25–35 (2003)
14. Quilan, J.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Francisco (1993)
15. R Project: Language and environment for statistical computing and graphics. R Foundation, <http://www.r-project.org>
16. Wang, X., Kerre, E.E.: Reasonable properties for the ordering of fuzzy quantities (I-II). *Fuzzy Sets and Systems* 118, 375–405 (2001)

A Fuzzy Logic Based Approach to Expressing and Reasoning with Uncertain Knowledge on the Semantic Web

Jidi Zhao¹, Harold Boley², and Weichang Du³

¹ Antai School of Management, Shanghai Jiao Tong University, Shanghai, China

² Institute for Information Technology, National Research Council of Canada, Fredericton, Canada

³ Faculty of Computer Science, University of New Brunswick, Fredericton, Canada

Abstract. The necessity of dealing with uncertain knowledge has arisen from Semantic Web applications in different areas. Handling uncertainty thus becomes one of the key research directions in the Semantic Web community. This paper introduces a fuzzy logic based approach which extends the classic Description Logic with Zadeh semantics to deal with uncertain knowledge about concepts and roles as well as instances of concepts and roles. Uncertain knowledge representation and the reasoning algorithm for consistency checking of a fuzzy knowledge base are addressed in detail. This paper also discusses complexity issues of the reasoning problem.

Keywords: Semantic web, Uncertain knowledge, Description logic, Fuzzy logic, Linear programming.

1 Introduction

The Semantic Web initiative aims at creating an extension to the current World Wide Web by developing logic-based standards and technologies that enable machines to understand the information on the Web, so that they can support richer knowledge inference and automate the performance of various tasks for human beings [2].

The current W3C standard for Semantic Web ontology languages, Web Ontology Language (OWL), is designed for use by applications that need to process the content of information instead of just presenting information to humans [15]. It facilitates greater machine interpretability of Web content than that supported by other Web languages such as XML, RDF, and RDF Schema (RDFS). This ability of OWL is enabled by its underlying knowledge representation formalism of Description Logics (DLs). DLs [17] are a family of logic-based formalisms designed to represent and reason about the conceptual knowledge of arbitrary domains. Elementary descriptions of DLs are atomic concepts and atomic roles. Complex concept descriptions and role descriptions can be built from the elementary descriptions according to construction rules. Different description languages of DLs are distinguished by the kind of concept and role constructors (such as conjunction, disjunction, and exists restriction) allowed in their description language and the kinds of axioms allowed in their terminologies. The basic

propositionally closed DL is \mathcal{ALC} in which the letters \mathcal{A} stand for attributive language and the letter \mathcal{C} for complement (negation of arbitrary concepts). Besides \mathcal{ALC} , other letters are used to indicate various DL extensions. For example, in the Description Logic \mathcal{ST} [7], \mathcal{S} is used for \mathcal{ALC} extended with transitive roles (R^+), and \mathcal{T} for inverse roles. DLs have a model-theoretic semantics, which is defined by interpreting concepts as sets of individuals and roles as sets of pairs of individuals. An interpretation I is a pair $I = (\Delta^I, \cdot^I)$ consisting of a domain Δ^I which is a non empty set and of an interpretation function \cdot^I which maps each individual x into an element of Δ^I ($x \in \Delta^I$), each concept C into a subset of Δ^I ($C^I \subseteq \Delta^I$) and each role R into a subset of $\Delta^I \times \Delta^I$ ($R^I \subseteq \Delta^I \times \Delta^I$). The semantics of complex concept and role descriptions can be found in [17]. Furthermore, a knowledge base (KB) in DLs consists of two parts: the terminological box (TBox T) and the assertional box (ABox A).

Uncertainty is an intrinsic feature of real-world knowledge, which is also reflected in the World Wide Web and the Semantic Web. Many concepts needed in knowledge modeling lack well-defined boundaries or, precisely defined criteria. Examples are the concepts of young, tall, and cold. The *Uncertainty Reasoning for the World Wide Web (URW3)* Incubator Group defined the challenge of representing and reasoning with uncertain information on the Web. According to the latest URW3 draft report, uncertainty is a term intended to encompass different forms of uncertain knowledge, including incompleteness, inconclusiveness, vagueness, ambiguity, and others [11]. The need to model and reason with uncertainty has been found in many different Semantic Web contexts, such as matchmaking in Web services [13], classification of genes in bioinformatics [18], multimedia annotation [17], and ontology learning [6].

Fuzzy Set Theory was first introduced by Zadeh [21] as an extension to the classic notion of a set to capture inherent vagueness (the lack of crisp boundaries of sets). Fuzzy Logic is a form of multi-valued logic derived from Fuzzy Set Theory to deal with reasoning that is approximate rather than precise. In Fuzzy Logic, the degree of truth of a statement can range between 0 and 1, and is not constrained to the two truth values $\{0, 1\}$ or $\{false, true\}$ as in classic predicate logic. Formally, a fuzzy set A with respect to a set of elements Ω (also called a universe) is characterized by a membership function $\mu_A(x)$ which assigns a value in the real unit interval $[0, 1]$ to each element x in Ω ($x \in \Omega$), notated as $\mu_A : \Omega \rightarrow [0, 1]$. $\mu_A(x)$, often written as $A(x)$, gives the degree of an element x belonging to the set A . Such degrees can be computed based on a membership function. A fuzzy relation R over two fuzzy sets A and B is similarly defined by a function $R : \Omega \times \Omega \rightarrow [0, 1]$.

Fuzzy Logic extends the Boolean operations defined on crisp sets and relations for fuzzy sets and fuzzy relations. These operations, e.g. complement, union, and intersection, are interpreted as mathematical functions over the unit interval $[0, 1]$. In the following, η, θ define the truth degrees of sets and relations, ranging between 0 and 1. The mathematical functions for fuzzy intersection are usually called t-norms ($t(\eta, \theta)$); those for fuzzy union are called s-norms ($s(\eta, \theta)$, a.k.a. t-conorms); and those for the fuzzy set complement are called negations ($\neg\eta$); These functions usually satisfy certain mathematical properties. The most widely known operations in the Fuzzy Logic family are Zadeh Logic, Lukasiewicz Logic, Product Logic, and Gödel Logic.

To deal with the ‘crisp limitation’ of classic DLs, considerable work has been carried out on integrating uncertain knowledge into DLs in recent years. The current literature generally follows two approaches. One is Probabilistic Logic based on Probability Theory; for example the work in [9] [10] [12]. The other is Fuzzy Logic and Fuzzy Sets; for example the work in [20] [19] [23]. A review and comparison of these works can be found in [22]. We presented a Norm-Parameterized Fuzzy Description Logic $fALCN$ and addressed the consistency checking problem in [23]. We use $f_{\mathfrak{N}}\mathcal{DL}$ to denote a Fuzzy Description Logic $f\mathcal{DL}$ with norm parameter \mathfrak{N} . Omitting the index \mathfrak{N} means the $f\mathcal{DL}$ is norm-parameterized. In the current paper, we follow the Fuzzy Sets and Fuzzy Logic approach and present the fuzzy Description Logic $f_Z\mathcal{SL}$. We call this fuzzy Description Logic $f_Z\mathcal{SL}$ as \mathcal{SL} is the underlying Description Logic and Z fixes the norms to Zadeh Logic. This paper is different from previous work due to the following features. First, the underlying classic DL \mathcal{SL} is a more expressive Description Logic which deals with fuzzy transitive roles and fuzzy inverse roles. Second, we combine Description Logic, Fuzzy Logic, and Linear Programming methods in the reasoning procedure. Last but not least, $f_Z\mathcal{SL}$ supports both fuzzy axioms and fuzzy assertions for uncertain knowledge representation and reasoning.

2 The Fuzzy DL $f_Z\mathcal{SL}$

$f_Z\mathcal{SL}$ extends the f_ZALCN DL with inverse roles, and transitive roles but excludes number restrictions. Due to space limitations, we refer interested readers to [23] for the syntax and semantics of complex concept descriptions as well as axioms and assertions for f_ZALC by specializing the t-norm to min and the s-norm to max. Here we simply list them in Tables 1 and 2, and then explain the expressiveness beyond f_ZALC . A fuzzy knowledge base in $f_Z\mathcal{SL}$ consists of two parts: the fuzzy terminological box consisting of a finite set of fuzzy axioms (TBox \mathcal{T}) and the fuzzy assertional box consisting of a finite set of fuzzy assertions (ABox \mathcal{A}). As shown in Table 2, a fuzzy axiom or fuzzy assertion is of the form $\alpha [l, u]$ with $0 \leq l \leq u \leq 1$, which is equivalent to the two inequalities $\alpha \geq l$ and $\alpha \leq u$. In what follows, we use these expressions as needed.

In classic DLs, a role R is symmetric iff for all $x, y \in \Delta^I$, $(Inv(R))^I(y, x) = R^I(x, y)$, where the role function $Inv(R)$ defines the inverse of a role. The same property holds for a fuzzy symmetric role. For example, the role *hasPart* is the inverse of the role *isPartOf*.

In classic DLs, a role R is transitive iff for all $x, y, z \in \Delta^I$, $R^I(x, y)$ and $R^I(y, z)$ imply $R^I(x, z)$. While in Fuzzy Logic, a fuzzy role R is transitive iff for all $x, y, z \in \Delta^I$, it satisfies the following inequality [3]:

$$R^I(x, z) \geq \sup_{y \in \Delta^I} t(R^I(x, y), R^I(y, z)) \quad (1)$$

where $t(\eta, \theta)$ denotes a general t-norm. Thus, in the case of θ Zadeh Logic, a transitive role satisfies:

$$R^I(x, z) \geq \sup_{y \in \Delta^I} \min(R^I(x, y), R^I(y, z)) \quad (2)$$

In order to make the following explanations easier, we introduce the role function $Trans(R)$ which specifies that R is transitive or $Inv(R)$ is transitive.

Table 1. Syntax and semantics of $f_Z\mathcal{S}\mathcal{I}$ constructors

Constructor	Syntax	Semantics
top concept	\top	$\top^I = 1$
bottom concept	\perp	$\perp^I = 0$
atomic negation	$\neg A$	$(\neg A)^I(x) = 1 - A^I(x)$
concept conjunction	$C \sqcap D$	$(C \sqcap D)^I = \min(C^I(x), D^I(x))$
concept disjunction	$C \sqcup D$	$(C \sqcup D)^I = \max(C^I(x), D^I(x))$
exists restriction	$\exists R.C$	$(\exists R.C)^I(x) = \sup_{y \in \Delta^I} \{\min(R^I(x, y), C^I(y))\}$
value restriction	$\forall R.C$	$(\forall R.C)^I(x) = \inf_{y \in \Delta^I} \{\max(1 - R^I(x, y), C^I(y))\}$
inverse role	$Inv(R)$	$(Inv(R))^I(y, x) = R^I(x, y)$

Table 2. Syntax and semantics of $f_Z\mathcal{S}\mathcal{I}$ axioms

Axioms	Syntax	Semantics
concept inclusion	$A \sqsubseteq C$	$\forall x \in \Delta^I, A^I(x) \leq C^I(x)$
concept definition	$A \equiv C$	$\forall x \in \Delta^I, A^I(x) = C^I(x)$
concept implication	$A \rightarrow C [l, u]$	$\forall x \in \Delta^I, C^I(x) \in \min(A^I(x), [l, u])$
transitive role	$Trans(R)$	$R^I(a, c) \geq \sup_{b \in \Delta^I} \min(R^I(a, b), R^I(b, c))$
concept assertion	$C(a) [l, u]$	$l \leq C^I(a) \leq u$
role assertion	$R(a, b) [l, u]$	$l \leq R^I(a, b) \leq u$
individual inequality	$a \neq b$	$a^I \neq b^I$

Now, we use some mathematical properties of Zadeh Logic to show that the following property is satisfied by a role value restriction $\forall R.C$ with $Trans(R)$.

Lemma 1. *Under Zadeh Logic, if $(\forall R.C)^I(x) \geq l$ ($l \in [0, 1]$) and R is transitive, then $(\forall R.(\forall R.C))^I(x) \geq l$ holds.*

Proof. $(\forall R.C)^I(x) \geq l$

$\xrightarrow{\text{Definition of semantics}}$

$$\inf_{z \in \Delta^I} \{\max(\neg R^I(x, z), C^I(z))\} \geq l$$

$\xrightarrow{\text{Equation } \square}$

$$\inf_{z \in \Delta^I} \inf_{y \in \Delta^I} \{\max(\neg(\min(R^I(x, y), R^I(y, z))), C^I(z))\} \geq l$$

$\xrightarrow{\text{De Morgan's Law}}$

$$\inf_{z \in \Delta^I} \inf_{y \in \Delta^I} \{\max(\max(\neg R^I(x, y), \neg R^I(y, z)), C^I(z))\} \geq l$$

$\xrightarrow{\text{Associativity}}$

$$\inf_{z \in \Delta^I} \inf_{y \in \Delta^I} \{\max(\neg R^I(x, y), \max(\neg R^I(y, z), C^I(z)))\} \geq l$$

$\xrightarrow{\text{Commutativity}}$

$$\inf_{y \in \Delta^I} \{\max(\neg R^I(x, y), \inf_{z \in \Delta^I} \max(\neg R^I(y, z), C^I(z)))\} \geq l$$

$\xrightarrow{\text{Definition of semantics}}$

$$\inf_{y \in \Delta^I} \{\max(\neg R^I(x, y), (\forall R.C)^I(y))\} \geq l$$

$\xrightarrow{\text{Definition of semantics}}$

$$(\forall R.(\forall R.C))^I(x) \geq l$$

However, in the cases of \leq , we cannot derive such a property for $(\forall R.C)^I(x)$ and $Trans(R)$.

Under Zadeh Logic, by applying the semantics of $\exists R.C$ and negation, it is easy to see that the following equivalence rules hold:

$$\forall a, b \in \Delta^I, \quad \neg\neg C \equiv C, \quad (3)$$

$$\neg\exists R.C \equiv \forall R.\neg C, \quad (4)$$

$$\neg\forall R.C \equiv \exists R.\neg C \quad (5)$$

Then, $(\exists R.C)^I(x) \leq u$

$$\xrightarrow{\text{Monotonicity}} \neg((\exists R.C)^I(x)) \geq 1 - u$$

$$\xrightarrow{\text{Equivalence } \color{red}{4}} (\forall R.(\neg C))^I(x) \geq 1 - u$$

$$\xrightarrow{\text{Lemma } \color{red}{1}} (\forall R.(\forall R.(\neg C)))^I(x) \geq 1 - u$$

$$\xrightarrow{\text{Monotonicity}} \neg(\forall R.(\forall R.(\neg C))^I(x)) \leq u$$

$$\xrightarrow{\text{Equivalence } \color{red}{5}} (\exists R.\neg(\forall R.(\neg C)))^I(x) \leq u$$

$$\xrightarrow{\text{Equivalence } \color{red}{5} \text{ and } \color{red}{3}} (\exists R.(\exists R.C))^I(x) \leq u$$

Therefore, the following property is satisfied with respect to a role exists restriction $\exists R.C$ and $Trans(R)$. Such a property cannot be inferred from the cases of \geq .

Lemma 2. *Under Zadeh Logic, if $(\exists R.C)^I(x) \leq u$ and R is transitive, then $(\exists R.(\exists R.C))^I(x) \leq u$ holds.*

Although we can show that such properties also hold under Product Logic and other logics, we neglect it here, as it is out of scope. We will soon see that these properties will be embodied in the fuzzy completion rules for the f_{ZSI} reasoning algorithm.

3 Reasoning Algorithm for Building a Fuzzy Tableau of f_{ZSI}

The reasoning algorithm that we will present is a fuzzy extension to the tableau method and tests the consistency of a knowledge base $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ by trying to construct a model of KB . A model of KB in our Fuzzy Description Logic f_{ZSI} is a fuzzy interpretation $I = (\Delta^I, \cdot^I)$. Similar to the classic DL, such a model has the shape of a forest, i.e., a collection of trees, with nodes corresponding to individuals, root nodes corresponding to named individuals, and edges corresponding to roles between individuals. Each node has a node label $\mathcal{L}(\text{individual})$, but different from classic DLs, each node in a f_{ZSI} tableau is labeled with a set of f_{ZSI} -concepts. Each element in the set consists of a pair of elements $\{\text{concept}, \text{constraint}\}$. The sets for all nodes are restricted to subsets of $sub(\mathcal{A})$, where $sub(\mathcal{A})$ is the set of sub-concepts of concepts that appear within an ABox \mathcal{A} . Furthermore, each edge is associated with an edge label $\mathcal{L}(\text{individual}_1, \text{individual}_2)$ which consists of a pair of elements $\{\text{role}, \text{constraint}\}$.

In [23], we explained the TBox processing procedure which consists of some preprocessing steps to deal with the fuzzy TBox before applying the reasoning algorithm. Those

steps are applicable to the $f_Z\mathcal{S}\mathcal{I}$ knowledge base. Therefore, we can assume all concepts C occurring in KB to be in *negation normal form* (NNF) and we only deal with unfoldable TBox after those preprocessing steps. However, due to the properties of a $f_Z\mathcal{S}\mathcal{I}$ knowledge base, the TBox processing procedure should include a couple of other steps. First, the TBox processing procedure transforms all the assertions in the fuzzy ABox and the fuzzy implication axioms in the fuzzy TBox with the form $\alpha_0 [l, u]$ into two expressions: $\alpha_0 \geq l$ and $\alpha_0 \leq u$. In order to keep our presentation simple and compact, in what follows, we use a general form $\alpha \text{ op } n$ where $\text{op} \in \{\geq, \leq\}$ and $n \in [0, 1]$ whenever applicable. Second, an $f_Z\mathcal{S}\mathcal{I}$ knowledge base may contain transitive role axioms and inverse roles. We know that if a role R is transitive, the inverse role of R is also transitive. Therefore, for each pair of $\text{Trans}(R)$ and $\text{Inv}(R)$, the procedure should also add an axiom $\text{Trans}(\text{Inv}(R))$. After the application of the TBox processing procedure, in what follows, we only have to consider a knowledge base in $f_Z\mathcal{S}\mathcal{I}$ only consists of fuzzy ABox assertions, a set of transitive role axioms, and a finite set of fuzzy implication axioms.

Next, we first give the definitions of fuzzy tableau, clash, and clash-free, and then present a decision procedure for the consistency checking problem for an $f_Z\mathcal{S}\mathcal{I}$ knowledge base.

Definition 1. *If $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is an $f_Z\mathcal{S}\mathcal{I}$ knowledge base, \mathbf{R}_A is the set of roles occurring in \mathcal{A} , together with their $\text{Inv}(R)$ s, a fuzzy tableau \mathfrak{T} for KB is defined to be a quadruple $(S, \mathcal{L}, \varepsilon, \mathcal{C})$ such that: S is a set of individuals, $\mathcal{L} : S \times \text{sub}(\mathcal{A}) \rightarrow [0, 1]$ maps each individual and a concept which is a subset of $\text{sub}(\mathcal{A})$ to the membership degree of the individual to that concept, $\varepsilon : \mathbf{R}_A \times S \times S \rightarrow [0, 1]$ maps each role in \mathbf{R}_A and a pair of individuals to the membership degree of the pair to the role, and \mathcal{C} is a set of constraints must be satisfied. For all $x, y \in S$, $A, C, D \in \text{sub}(\mathcal{A})$, $R \in \mathbf{R}_A$ and $n \in [0, 1]$, it holds that:*

1. For any $x \in S$, $\{x : \perp = 0\}$ and $\{x : \top = 1\} \in \mathcal{L}(x)$.
2. If $\{x : \neg(A) \text{ op } n\} \in \mathcal{L}(x)$, then $\{x : A \text{ op } 1 - n\} \in \mathcal{L}(x)$.
3. If $\{x : C \sqcap D \geq n\} \in \mathcal{L}(x)$, then $\{x : C \geq n\} \in \mathcal{L}(x)$ and $\{x : D \geq n\} \in \mathcal{L}(x)$.
4. If $\{x : C \sqcup D \leq n\} \in \mathcal{L}(x)$, then $\{x : C \leq n\} \in \mathcal{L}(x)$ and $\{x : D \leq n\} \in \mathcal{L}(x)$.
5. If $\{x : C \sqcap D \leq n\} \in \mathcal{L}(x)$, then $\{x : C \leq n_1\} \in \mathcal{L}(x)$, $\{x : D \leq n_2\} \in \mathcal{L}(x)$, and $n = \min(n_1, n_2)$ for some n_1, n_2 .
6. If $\{x : C \sqcup D \geq n\} \in \mathcal{L}(x)$, then $\{x : C \geq n_1\} \in \mathcal{L}(x)$, $\{x : D \geq n_2\} \in \mathcal{L}(x)$, and $n = \max(n_1, n_2)$ for some n_1, n_2 .
7. If $\{x : \exists R.C \geq n\} \in \mathcal{L}(x)$, then there exists $y \in S$ such that $\{\langle x, y \rangle : R \geq n\} \in \varepsilon(R)$ and $\{y : C \geq n\} \in \mathcal{L}(y)$.
8. If $\{x : \forall R.C \leq n\} \in \mathcal{L}(x)$, then there exists $y \in S$ such that $\{\langle x, y \rangle : R \geq 1 - n\} \in \varepsilon(R)$ and $\{y : C \leq n\} \in \mathcal{L}(y)$.
9. If $\{x : \exists R.C \leq n\} \in \mathcal{L}(x)$, then $\{\langle x, y \rangle : R \leq n_1\} \in \varepsilon(R)$, $\{y : C \leq n_2\} \in \mathcal{L}(y)$, and $n = \min(n_1, n_2)$ for some n_1, n_2 .
10. If $\{x : \forall R.C \geq n\} \in \mathcal{L}(x)$, then $\{\langle x, y \rangle : R \leq 1 - n_1\} \in \varepsilon(R)$, $\{y : C \geq n_2\} \in \mathcal{L}(y)$, and $n = \max(1 - n_1, n_2)$ for some n_1, n_2 .
11. $\{\langle x, y \rangle : R \text{ op } n\} \in \varepsilon(R)$ iff $\{\langle y, x \rangle : \text{Inv}(R) \text{ op } n\} \in \varepsilon(R)$.
12. If $\{x : \forall R.C \geq n\} \in \mathcal{L}(x)$ and $\text{Trans}(R)$, then $\{\langle x, y \rangle : R \leq 1 - n_1\} \in \varepsilon(R)$, $\{y : \forall R.C \geq n_2\} \in \mathcal{L}(y)$, and $n = \max(1 - n_1, n_2)$ for some n_1, n_2 .

13. If $\{x : \exists R.C \leq n\} \in \mathcal{L}(x)$ and $\text{Trans}(R)$, then $\{< x, y > : R \leq n_1\} \in \varepsilon(R)$, $\{y : \exists R.C \leq n_2\} \in \mathcal{L}(y)$, and $n = \min(n_1, n_2)$ for some n_1, n_2 .
14. If $\{A \rightarrow C \geq n\} \in \mathcal{T}$ and $\{x : A \geq n_1\} \in \mathcal{L}(x)$, then $\{x : C \geq n_2\} \in \mathcal{L}(x)$ and $n_2 = \min(n, n_1)$, for any $x \in S$.
15. If $\{A \rightarrow C \leq n\} \in \mathcal{T}$ and $\{x : A \leq n_1\} \in \mathcal{L}(x)$, then $\{x : C \leq n_2\} \in \mathcal{L}(x)$ and $n_2 = \min(n, n_1)$, for any $x \in S$.

In [23], we defined the semantics $(C \sqcap D)^I$ as $t(C^I(x), D^I(x))$ for various t-norms. For the case of Zadeh Logic, we have that if $(C \sqcap D)^I(x) \geq n$, then $C^I(x) = n_C$, $D^I(x) = n_D$, and $\min(n_C, n_D) \geq n$. In this definition, we can draw a further conclusion based on the properties of the min norm that $C^I(x) = n_C \geq n$ and $D^I(x) = n_D \geq n$. Similar extensions are conducted on other f_{ZSI} concepts and roles.

Definition 2. Let \mathcal{A} be an extended f_{ZSI} ABox, \mathcal{A} contains a clash if only if one of the following situations occurs:

1. $\{\perp(a) \neq 0\} \subseteq \mathcal{A}$
2. $\{\top(a) \neq 1\} \subseteq \mathcal{A}$
3. $\{\alpha \leq n_1, \alpha \geq n_2\} \subseteq \mathcal{A}$ and $n_1 < n_2$
4. there is no solution for the constraint system of inequations \mathcal{C}

\mathcal{A} is called clash-free if it does not contain any clash.

Similar to the tableau algorithm presented by Horrocks et al. [8], our algorithm works on building a fuzzy tableau for an f_{ZSI} knowledge base which may be a completion-forest since the ABox might contain several named individuals with arbitrary edges connecting them. Each node x is labeled with a set $\mathcal{L}(x) = \{\{x : C_1 \text{ op } n_1\}, \dots, \{x : C_m \text{ op } n_m\}\}$ ($m \geq 1$) and a constraint set $\mathcal{C}(x) = \{\{x_{C_1} \text{ op } n_1\}, \dots, \{x_{C_m} \text{ op } n_m\}\}$, where $C_i \in \text{sub}(\mathcal{A})$, $x_{C_i}, n_i \in [0, 1]$, $1 \leq i \leq m$, and $\text{op} \in \{\geq, \leq\}$. Each edge $< x, y >$ is labeled with a set $\mathcal{L}(x, y) = \{[x, y] : R \text{ op } n\}$ and a constraint in the set $\mathcal{C}(x, y) = \{x_R \text{ op } n\}$, where R are roles occurring in \mathcal{A} .

We adapt the conjugation concept in [19] to represent pairs of fuzzy assertions that form a contradiction. Let α be a SI assertion, two fuzzy assertions ($\alpha \geq n_1$ and $\alpha \leq n_2$) conjugate with each other if $n_1 > n_2$. For a given fuzzy assertion, its conjugated assertion is not unique, and in fact, infinite. For example, both $\{[x, y] : R \leq 0.5\}$ and $\{[x, y] : R \leq 0.3\}$ conjugate with the fuzzy assertion $\{[x, y] : R \geq 0.6\}$.

Let us recall some notations used in [7]. If nodes x and y are connected by an edge $< x, y >$ with $\{R \text{ op } n\} \in \mathcal{L}(x, y)$, then y is called an R_n -successor of x and x is called an R_n -predecessor of y . Ancestor is the transitive closure of predecessor. If y is an R_n -successor or an $(\text{Inv}(R))_n$ -predecessor of x , then y is called an R_n -neighbor of x . An expressive DL such as f_{ZSI} which allows transitive roles and inverse roles may lead to nontermination as the fuzzy completion rules can introduce new concepts that are the same size as the decomposed concept. Our algorithm for the consistency checking of an f_{ZSI} knowledge base follows the dynamic blocking presented in [7] and uses it to guarantee the termination of the reasoning algorithm. In dynamic blocking, blocked nodes are allowed to be dynamically established and broken as the expansion progresses, and expand role value restriction and role exists restriction concepts.

This dynamic blocking strategy is crucial in the presence of inverse roles since information might be propagated up the completion-forest and affect other branches. For example, consider the nodes x, y and z , the edges $\langle x, y \rangle$ and $\langle x, z \rangle$. Suppose x blocks y . In the presence of inverse roles it is possible that z adds information to node x , although z is a successor of x . In that case the block on y must be broken. A node x is *blocked* if for some ancestor y , y is blocked or $\mathcal{L}(x) = \mathcal{L}(y)$. *Dynamic blocking* uses the notions of *directly blocked* and *indirectly blocked* nodes. If a blocked node x 's predecessor is blocked, x is called *indirectly blocked*. A blocked node x is called *directly blocked* if it has a unique ancestor y such that $\mathcal{L}(x) = \mathcal{L}(y)$.

Now, for an expanded f_{ZSI} ABox \mathcal{A} with a set of transitive role axioms and a set of fuzzy implication axioms, the algorithm initializes a forest to contain (1) root nodes, for each individual x occurring in \mathcal{A} , the root node x is labeled with $\mathcal{L}(x) = \{x : C \text{ op } n\}$ and $\mathcal{C}(x) = \{x_C \text{ op } n\}$ for each assertion of the form $C(x) \text{ op } n$ in \mathcal{A} , and (2) edges, each edge $\langle x, y \rangle$ corresponds to an assertion $R(x, y) \text{ op } n$ in \mathcal{A} with R be an atomic role or an inverse role and is labeled with $\mathcal{L}(x, y) = \{[x, y] : R \text{ op } n\}$ and $\mathcal{C}(x, y) = \{x_R \text{ op } n\}$. If an assertion is of the form $Inv(P)(x, y) \text{ op } n$, the corresponding edge is also labeled with $\mathcal{L}(x, y) = \{[y, x] : P \text{ op } n\}$ and $\mathcal{C}(x, y) = \{x_P \text{ op } n\}$. The completion forest is then expanded by repeatedly applying the following fuzzy completion rules in Table 3. The completion forest is complete when a clash is detected, or none of the fuzzy completion rules are applicable.

The algorithm stops when a clash occurs; KB is consistent iff the completion rules can be applied in such a way that they yield a complete and clash-free completion forest, and KB is inconsistent otherwise.

Table 3. Fuzzy Completion Rules for f_{ZSI}

f_{ZSI} Fuzzy Completion Rules
<p>$\neg \geq$-rule Condition: $\{x : (\neg A) \geq n\} \in \mathcal{L}(x)$ and $\{x : A \leq 1 - n\} \notin \mathcal{L}(x)$ Action: $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{x : A \leq 1 - n\}$ and $\mathcal{C}(x) \rightarrow \mathcal{C}(x) \cup \{x_A \leq (1 - n)\}$</p>
<p>$\neg \leq$-rule Condition: $\{x : (\neg A) \leq n\} \in \mathcal{L}(x)$ and $\{x : A \geq 1 - n\} \notin \mathcal{L}(x)$ Action: $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{x : A \geq 1 - n\}$ and $\mathcal{C}(x) \rightarrow \mathcal{C}(x) \cup \{x_A \geq (1 - n)\}$</p>
<p>$\sqcap \geq$-rule Condition: $\{x : (C_1 \sqcap C_2) \geq n\} \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{\{x : C_1 \geq n\}, \{x : C_2 \geq n\}\} \not\subseteq \mathcal{L}(x)$ Action: $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\{x : C_1 \geq n\}, \{x : C_2 \geq n\}\}$ and $\mathcal{C}(x) \rightarrow \mathcal{C}(x) \cup \{x_{C_1} \geq n, x_{C_2} \geq n\}$</p>
<p>$\sqcap \leq$-rule Condition: $\{x : (C_1 \sqcap C_2) \leq n\} \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{\{x : C_1 \leq n\}, \{x : C_2 \leq n\}\} \cap \mathcal{L}(x) = \emptyset$ Action: $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\{x : C_1 \leq x_1\}, \{x : C_2 \leq x_2\}\}$ and $\mathcal{C}(x) \rightarrow \mathcal{C}(x) \cup \{x_{C_1} \leq x_1, x_{C_2} \leq x_2, x_1 + x_2 = 1 + n, x_1 \geq y, x_2 \geq 1 - y, y \in \{0, 1\}, x_1 \in [0, 1], x_2 \in [0, 1]\}$</p>
<p>$\sqcup \geq$-rule Condition: $\{x : (C_1 \sqcup C_2) \geq n\} \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{\{x : C_1 \geq n\}, \{x : C_2 \geq n\}\} \cap \mathcal{L}(x) = \emptyset$ Action: $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\{x : C_1 \geq x_1\}, \{x : C_2 \geq x_2\}\}$ and $\mathcal{C}(x) \rightarrow \mathcal{C}(x) \cup \{x_{C_1} \geq x_1, x_{C_2} \geq x_2, x_1 + x_2 = n, x_1 \leq y, x_2 \leq 1 - y, y \in \{0, 1\}, x_1 \in [0, 1], x_2 \in [0, 1]\}$</p>

Table 3. (continued)

f_{zSI} Fuzzy Completion Rules
<p>\sqcup_{\leq}-rule</p> <p>Condition: $\{x : (C_1 \sqcup C_2) \leq n\} \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{\{x : C_1 \leq n\}, \{x : C_2 \leq n\}\} \not\subseteq \mathcal{L}(x)$</p> <p>Action: $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\{x : C_1 \leq n\}, \{x : C_2 \leq n\}\}$ and $\mathcal{C}(x) \rightarrow \mathcal{C}(x) \cup \{x_{C_1} \leq n, x_{C_2} \leq n\}$</p>
<p>\exists_{\geq}-rule</p> <p>Condition: $\{x : (\exists R.C) \geq n\} \in \mathcal{L}(x)$, x is not blocked, and x has no R_n-neighbor y</p> <p>Action: create a new node y with $\mathcal{L}(x, y) = \{\{[x, y] : R \geq n\}\}$, $\mathcal{L}(y) = \{\{y : C \geq n\}\}$, $\mathcal{C}(x, y) = \{x_R \geq n\}$, and $\mathcal{C}(y) = \{x_C \geq n\}$</p>
<p>\exists_{\leq}-rule</p> <p>Condition: $\{x : (\exists R.C) \leq n\} \in \mathcal{L}(x)$, x is not indirectly blocked, and x has an R_{n1R}-neighbor y with $\{[x, y] : R \text{ op } n1\} \in \mathcal{L}(x, y)$ and $\{y : C \leq n\} \notin \mathcal{L}(y)$.</p> <p>Action: $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\{y : C \leq n\}\}$, if $\{[x, y] : R \text{ op } n1\}$ conjugates with $\{[x, y] : R \leq n\}$, then $\mathcal{C}(y) \rightarrow \mathcal{C}(y) \cup \{x_C \leq n\}$, else $\mathcal{C}(y) \rightarrow \mathcal{C}(y) \cup \{x_C \leq n, n1 > n\}$</p>
<p>\forall_{\geq}-rule</p> <p>Condition: $\{x : (\forall R.C) \geq n\} \in \mathcal{L}(x)$, x is not indirectly blocked and x has an R_{n1R}-neighbor y with $\{y : C \geq n\} \notin \mathcal{L}(y)$</p> <p>Action: $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\{y : C \geq n\}\}$, if $\{[x, y] : R \text{ op } n1\}$ conjugates with $\{[x, y] : R \leq (1 - n)\}$, then $\mathcal{C}(y) \rightarrow \mathcal{C}(y) \cup \{x_C \geq n\}$, else $\mathcal{C}(y) \rightarrow \mathcal{C}(y) \cup \{x_C \geq n, n1 > 1 - n\}$</p>
<p>\forall_{\leq}-rule</p> <p>Condition: $\{x : (\forall R.C) \leq n\} \in \mathcal{L}(x)$, x is not blocked, x has no R_n-neighbor y, and $\{y : C \leq n\} \in \mathcal{L}(y)$</p> <p>Action: create a new node y with $\mathcal{L}(x, y) = \{\{[x, y] : R \geq (1 - n)\}\}$, $\mathcal{L}(y) = \{\{y : C \leq n\}\}$, $\mathcal{C}(x, y) = \{x_R \geq (1 - n)\}$, and $\mathcal{C}(y) = \{x_C \leq n\}$</p>
<p>$\exists_{\leq,+}$-rule</p> <p>Condition: $\{x : (\exists R.C) \leq n\} \in \mathcal{L}(x)$, $Trans(R)$, x is not indirectly blocked, and x has an R_{n1R}-neighbor y with $\{y : (\exists R.C) \leq n\} \notin \mathcal{L}(y)$</p> <p>Action: $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\{y : (\exists R.C) \leq n\}\}$, if $\{[x, y] : R \text{ op } n1\}$ conjugates with $\{[x, y] : R \leq n\}$, then $\mathcal{C}(y) \rightarrow \mathcal{C}(y) \cup \{x_{\exists R.C} \leq n\}$, else $\mathcal{C}(y) \rightarrow \mathcal{C}(y) \cup \{x_{\exists R.C} \leq n, n1 > n\}$</p>
<p>$\forall_{\geq,+}$-rule</p> <p>Condition: $\{x : (\forall R.C) \geq n\} \in \mathcal{L}(x)$, $Trans(R)$, x is not indirectly blocked, and x has an R_{n1R}-neighbor y with $\{y : (\forall R.C) \geq n\} \notin \mathcal{L}(y)$</p> <p>Action: $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{\{y : (\forall R.C) \geq n\}\}$, if $\{[x, y] : R \text{ op } n1\}$ conjugates with $\{[x, y] : R \leq (1 - n)\}$, then $\mathcal{C}(y) \rightarrow \mathcal{C}(y) \cup \{x_{\forall R.C} \geq n\}$, else $\mathcal{C}(y) \rightarrow \mathcal{C}(y) \cup \{x_{\forall R.C} \geq n, n1 > 1 - n\}$</p>
<p>\rightarrow_{\geq}-rule</p> <p>Condition: $\{A \rightarrow C \geq n\} \in \mathcal{T}$, $\{x : A \geq n1\} \in \mathcal{L}(x)$</p> <p>Action: $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\{x : D \geq n2\}\}$ and $\mathcal{C}(x) \rightarrow \mathcal{C}(x) \cup \{x_D \geq n2, n2 = \min(n, n1)\}$</p>
<p>\rightarrow_{\leq}-rule</p> <p>Condition: $\{A \rightarrow C \leq n\} \in \mathcal{T}$, $\{x : A \leq n1\} \in \mathcal{L}(x)$</p> <p>Action: $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{\{x : D \leq n2\}\}$ and $\mathcal{C}(x) \rightarrow \mathcal{C}(x) \cup \{x_D \leq n2, n2 = \min(n, n1)\}$</p>

Example 1. Consider a fuzzy knowledge base $KB = \{ CP \rightarrow \exists hP.CP [0.5, 1], CP(P002) [0.6, 1], (\exists hP.CP)(P002) [0, 0.4] \}$ where we abbreviate the concept *CancerPatient* and the role *hasFirstDegreeRelatives* by CP and hP , respectively. The knowledge base describes that the truth degree for a first-degree relative of a cancer patient also being a cancer patient is greater than or equal to 0.5. Person $P002$ is a cancer patient with certainty greater than 0.6 and the possibility that one of $P002$'s first-degree relative is also a cancer patient is less than or equal to 0.4. The query is that whether KB is consistent or not.

First, because of the fuzzy concept implication axiom, $\{\exists hP.CP(P002) [0.5, 1]\}$ is added to \mathcal{A} . Next, we can initialize the fuzzy tableau by creating a node $P002$ and label it with $\mathcal{L}(P002) = \{\{P002 : CP \geq 0.6\}, \{P002 : \exists hP.CP \geq 0.5\}, \{P002 : \exists hP.CP \leq 0.4\}\}$ and $\mathcal{C}(P002) = \{x_{CP} \geq 0.6, x_{\exists hP.CP} \geq 0.5, x_{\exists hP.CP} \leq 0.4\}$. Since both $\{P002 : \exists hP.CP \geq 0.5\}$ and $\{P002 : \exists hP.CP \leq 0.4\}$ are contained in the fuzzy tableau, the reasoning algorithm obviously detects a clash. Therefore, it stops the application of any fuzzy completion rule and returns that KB is inconsistent.

Next, let us look at an example for the $\forall_{\geq, +}$ -rule.

Example 2. Consider there are two assertions in a fuzzy knowledge base: $(\forall hasFriend.Student)(John) [0.75, 1]$ and $hasFriend(John, Mary) [0.7, 1]$ where *hasFriend* is a transitive role.

Following the preprocessing steps, we have $\{John : (\forall hasFriend.Student) \geq 0.75\} \in \mathcal{L}(John)$ and $\{[John, Mary] : hasFriend \geq 0.7\}$. Since $\{[John, Mary] : hasFriend \geq 0.7\}$ conjugates with $\{[John, Mary] : hasFriend \leq 0.25\}$, the $\forall_{\geq, +}$ -rule is applicable, thus $\{Mary : (\forall hasFriend.Student) \geq 0.75\}$ is added to $\mathcal{L}(Mary)$.

We can see from Table 3 that all these fuzzy completion rules are based on the properties and the semantics of $fz\mathcal{SL}$ concepts. Notice that since we assume all concepts to be in their negation normal form, the fuzzy concept negation rule only applies to concept names.

Let us take a second look at the \sqcup_{\geq} -rule and the \sqcap_{\leq} -rule. The \sqcup_{\geq} -rule generates several new constraints $\{x_1 + x_2 = n, x_1 \leq y, x_2 \leq 1 - y, y \in \{0, 1\}, x_1 \in [0, 1], x_2 \in [0, 1]\}$. We can see that y is an integer variable with value of 0 or 1. When $y = 0$, we have $x_1 = 0, x_2 = n$, and thus $\{x_{C_1} \geq 0, x_{C_2} \geq n\}$; while $y = 1$, we have $x_1 = n, x_2 = 0$, and thus $\{x_{C_1} \geq n, x_{C_2} \geq 0\}$. These two cases are actually representing the or-branch of the concept disjunction rule in classic DL. That is, the $\{0, 1\}$ integer variable y enable the simulation of or-branching. Furthermore, by the introduction of the variable y , we also transform the non-linear constraint $max(x_1, x_2) \geq n$ into a set of linear constraints. Similar conclusions can be drawn on the \sqcap_{\leq} -rule. Now we can see that all the fuzzy completion rules in Table 3 generate only linear constraints, therefore, the resulted constraint set for any node or edge is a linear constraint set. Such a property makes it possible for the reasoning algorithm to call some external *Linear Programming* solver to solve the constraint set.

Here is another example to explain how the reasoning algorithm determines the consistency of a knowledge base.

Example 3. Consider the following fuzzy knowledge base $KB = \{Trans(R), C(a) [0.7, 1], D(b) [0.8, 1], R(a, b) [0.6, 1], R(b, c) [0.7, 1], (\exists Inv(R).C \cap \exists Inv(R).D)(c) [0, 0.5]\}$. We want to check the consistency of the knowledge base.

With $Trans(R)$ and $Inv(R)$, we have $Trans(Inv(R))$. The fuzzy tableau is initialized as shown in Figure 11

Next, since $\{c : (\exists Inv(R).C \cap \exists Inv(R).D) \leq 0.5\} \in \mathcal{L}(c)$, the \sqcap_{\leq} -rule is triggered, the reasoning algorithm adds $\{c : (\exists Inv(R).C) \leq x_1\}$ and $\{c : (\exists Inv(R).D) \leq x_2\}$ to $\mathcal{L}(c)$, adds $\{x_{(\exists Inv(R).C)(c)} \leq x_1, x_{(\exists Inv(R).D)(c)} \leq x_2, x_1 + x_2 = 1 + 0.5, x_1 \geq y, x_2 \geq 1 - y, y \in \{0, 1\}, x_1 \in [0, 1], x_2 \in [0, 1]\}$ to $\mathcal{C}(c)$.

Next, since $\{c : (\exists Inv(R).C) \leq x_1\} \in \mathcal{L}(c)$, $\{c : (\exists Inv(R).D) \leq x_2\} \in \mathcal{L}(c)$, and we have $[b, c] : R \geq 0.7$, the \exists_{\leq} -rule is applicable, thus the reasoning algorithm adds $\{b : C \leq x_1\}$ and $\{b : D \leq x_2\}$ to $\mathcal{L}(b)$, adds $\{x_{C(b)} \leq x_1, x_{D(b)} \leq x_2, x_1 < 0.7, x_2 < 0.7\}$ to $\mathcal{C}(b)$. Note that the constraints $x_1 < 0.7$ and $x_2 < 0.7$ are added because of conjugation.

Next, since $\{c : (\exists Inv(R).C) \leq x_1\} \in \mathcal{L}(c)$, $\{c : (\exists Inv(R).D) \leq x_2\} \in \mathcal{L}(c)$, we have $[b, c] : R \geq 0.7$ and $Trans(Inv(R))$, the $\exists_{\leq, +}$ -rule is also applicable, thus the reasoning algorithm adds $\{b : (\exists Inv(R).C) \leq x_1\}$ and $\{b : (\exists Inv(R).D) \leq x_2\}$ to $\mathcal{L}(b)$, adds $\{x_{(\exists Inv(R).C)(b)} \leq x_1, x_{(\exists Inv(R).D)(b)} \leq x_2\}$ to $\mathcal{C}(b)$.

Next, since $\{b : (\exists Inv(R).C) \leq x_1\} \in \mathcal{L}(b)$, $\{b : (\exists Inv(R).D) \leq x_2\} \in \mathcal{L}(b)$, and we have $[a, b] : R \geq 0.6$, the \exists_{\leq} -rule is also applicable, thus the reasoning algorithm adds $\{a : C \leq x_1\}$ and $\{a : D \leq x_2\}$ to $\mathcal{L}(a)$, adds $\{x_{C(b)} \leq x_1, x_{D(b)} \leq x_2, x_1 < 0.6, x_2 < 0.6\}$ to $\mathcal{C}(a)$.

Now the fuzzy tableau is shown in Figure 12. Together with the default variable constraints, the reasoning algorithm forms the following constraint set:

$$\text{subject to } \left\{ \begin{array}{l} x_{C(a)} \geq 0.7, x_{D(b)} \geq 0.8 \\ x_{R(a,b)} \geq 0.6, x_{R(b,c)} \geq 0.7 \\ x_{(\exists Inv(R).C \cap \exists Inv(R).D)(c)} \leq 0.5 \\ x_{(\exists Inv(R).C)(c)} \leq x_1, x_{(\exists Inv(R).D)(c)} \leq x_2 \\ x_1 + x_2 = 1 + 0.5 \\ x_1 \geq y, x_2 \geq 1 - y \\ x_{C(b)} \leq x_1, x_{D(b)} \leq x_2 \\ x_1 < 0.7, x_2 < 0.7 \\ x_{C(a)} \leq x_1, x_{D(a)} \leq x_2 \\ x_1 < 0.6, x_2 < 0.6 \\ x_{(\exists Inv(R).C)(b)} \leq x_1, x_{(\exists Inv(R).D)(b)} \leq x_2 \\ x_{C(a)}, x_{D(b)}, x_{R(a,b)}, x_{R(b,c)} \in [0, 1] \\ x_{(\exists Inv(R).C \cap \exists Inv(R).D)(c)} \in [0, 1] \\ x_{(\exists Inv(R).C)(c)}, x_{(\exists Inv(R).D)(c)} \in [0, 1] \\ y \in \{0, 1\} \\ x_1, x_2, x_{C(b)}, x_{D(b)} \in [0, 1] \\ x_{(\exists Inv(R).C)(b)}, x_{(\exists Inv(R).D)(b)} \in [0, 1] \end{array} \right.$$

Using a *Linear Programming* solver, e.g., the GLPK solver [4], it is easy to show that the constraint set is unsolvable. Therefore, the fuzzy knowledge base is inconsistent.

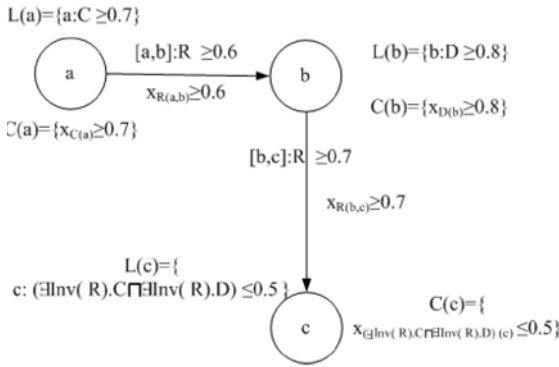


Fig. 1. The initial fuzzy tableau of example 3

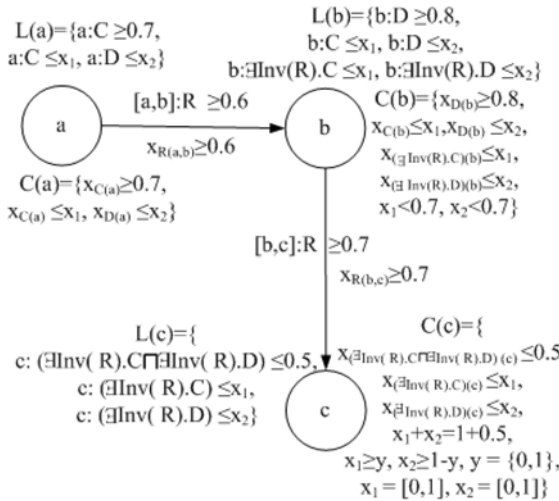


Fig. 2. The extended fuzzy tableau of example 3

Through this example, it is shown that the consistency check of a knowledge base can be reduced to a problem of constraints solving in linear programming. The constraints solving can be processed either at the end of the reasoning procedure when no further fuzzy completion rules are applicable, or after each application of a completion rule. The advantage of the later case is that, in some situations, the computation effort could be saved when the constraints solver can identify unsolvable constraints sets earlier in the reasoning process. However, in other situations, since calling an external solver is time consuming, frequent calls will severely affect the overall performance. In the former case, we only have to call the external solver once. In addition, we can apply some optimization strategies such as trivial clash detection and individual groups to improve the performance.

In the following, we discuss complexity issues of the reasoning problem in $f_Z\mathcal{SL}$. The lower bound complexity of the consistency checking problem in $f_Z\mathcal{SL}$ is EXP-time. Horrocks and Sattler showed that the consistency checking problem for an \mathcal{SL} knowledge base is complete for EXP [7]. Since every \mathcal{SL} knowledge base is a special case of $f_Z\mathcal{SL}$, the consistency checking problem in $f_Z\mathcal{SL}$ inherits \mathcal{SL} 's time complexity as its lower bound. The overhead for the consistency checking algorithm in $f_Z\mathcal{SL}$ is approximately a linear factor ($O(n)$) of the time complexity of the conventional \mathcal{SL} algorithm. It results from applying the completion rules, adding constraints to the constraint set, and solving the mixed integer linear programming problem on the application of each fuzzy completion rule. Each completion rule in the \mathcal{SL} reasoning algorithm is extended into at most two fuzzy completion rules in $f_Z\mathcal{SL}$. Furthermore, each fuzzy completion rule includes the extra step of adding constraints to the constraint set. Assuming τ_3 is the computational complexity of \mathcal{SL} 's consistency checking algorithm, the upper bound complexity for the fuzzy tableau construction part is $O(4 * \tau_3)$. Next, let τ_4 be the time complexity of solving a mixed integer linear programming problem, where some of the unknown variables are required to be integers from $\{0, 1\}$. It is well-known that a mixed integer linear programming problem is NP-complete [5]. Solving a linear program is bounded by $O(m^3 * L)$ where L is the maximum 'bit size' of coefficients in the linear program and m is the number of variables [14]. Solving a binary mixed integer problem is bounded by $O(2^q * m^3 * L)$ where q is the number of integer variables. In our case, q and m are bounded by the size of the knowledge base, thus τ_4 is $O(2^n * n^3 * L)$. Assuming $p(n)$ is $\Omega(n^{1+\epsilon})$, then $O(2^n * n^3 * L)$ is less than $O(2^{p(n)})$ when n is large; therefore, the upper bound of the time complexity for the consistency checking in $f_Z\mathcal{SL}$ is $O(4 * \tau_3 + n * \tau_4) = O(n * \tau_3)$, where n is the size of the underlying knowledge base. If the reasoning algorithm processes mixed integer linear programming only once at the end, when no more fuzzy completion rule is applicable, the time complexity of $f_Z\mathcal{SL}$ becomes $O(\tau_3)$.

4 Conclusions

In this paper, we address the fuzzy instance entailment problem with respect to a fuzzy knowledge base and then present a fuzzy extension to the expressive Description Logic \mathcal{SL} based on Zadeh Logic and the residual R-implication.

For real-world applications where a knowledge base is considered as a means to store information (both precise and imprecise) about individuals, usually more complex inferences other than consistency checking are required. For example, users may want to pose a query like "Given a knowledge base, what's the certainty of an assertion?". Another kind of query can be "How many individuals belong to a given concept description with a confidence greater than 0.5, and what are they?" We describe the former query as an instance range entailment problem and the later as an f-retrieval problem. However, due to space limitations, the reasoning methods for these problems are omitted in this paper.

A prototype reasoner using SWI-Prolog and GLPK has been under implementation based on the \mathcal{ALC} reasoner ALCAS [16]. It currently supports functionalities to check consistency, fuzzy instance entailment and f-retrieval of a fuzzy $f_Z\mathcal{ALC}$ knowledge

base. Part of our ongoing work considers further development of the reasoner to support other reasoning problems as well as more expressivity in the fuzzy knowledge base.

As we pointed out in Section 2 the properties for transitive roles and value restrictions also hold under Product Logic. Therefore, another direction of future work is to investigate the reasoning algorithms for expressive fuzzy Description Logics under norms from other logics in the family of Fuzzy Logics.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2003)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. *Scientific American* 284, 34–44 (2001)
3. Diaz, S., De Baets, B., Montes, S.: General results on the decomposition of transitive fuzzy relations. *Fuzzy Optimization and Decision Making* 9, 1–29 (2010)
4. GLPK: GNU linear programming kit. Technical Report (2008), <http://gnuwin32.sourceforge.net/packages/glpk.htm>
5. Lenstra Jr., H.W.: Integer programming with a fixed number of variables. *Mathematics of Operations Research* 8(4), 538–548 (1983)
6. Haase, P., Völker, J.: Ontology learning and reasoning - dealing with uncertainty and inconsistency. In: *Proceedings of Uncertainty Reasoning for the Semantic Web*, pp. 45–55 (2005)
7. Horrocks, I., Sattler, U.: A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation* 9, 385–410 (1999)
8. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with individuals for the description logic *SHIQ*. In: McAllester, D. (ed.) *CADE 2000*. LNCS, vol. 1831, pp. 482–496. Springer, Heidelberg (2000)
9. Jaeger, M.: Probabilistic reasoning in terminological logics. In: *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 1994)*, pp. 305–316 (1994)
10. Koller, D., Levy, A., Pfeffer, A.: P-classic: A tractable probabilistic description logic. In: *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI 1997)*, pp. 390–397 (1997)
11. Laskey, K.J., Laskey, K.B., Costa, P.C.G., Kokar, M.M., Martin, T., Lukasiewicz, T.: W3C incubator group report. Technical Report, W3C (March 05, 2008), <http://www.w3.org/2005/Incubator/urw3/>,
12. Lukasiewicz, T.: Fuzzy description logic programs under the answer set semantics for the semantic web. *Fundamenta Informaticae* 82, 289–310 (2008)
13. Martin-Recuerda, F., Robertson, D.: Discovery and uncertainty in semantic web services. In: *Proceedings of Uncertainty Reasoning for the Semantic Web*, vol. 188 (2005)
14. Matousek, J., G'artner, B.: *Understanding and Using Linear Programming*. Springer, Heidelberg (2007)
15. McGuinness, D.L., van Harmelen, F.: Owl web ontology language overview (2004), <http://www.w3.org/TR/owl-features/>
16. Spencer, B.: ALCAS: An ALC Reasoner for CAS (2006), <http://www.cs.unb.ca/bspender/cs6795swt/alcas.prolog>
17. Stamou, G., van Ossenbruggen, J., Pan, J.Z., Schreiber, G.: Multimedia annotations on the semantic web. *IEEE MultiMedia* 13, 86–90 (2006)
18. Stevens, R., Aranguren, M.E., Wolstencroft, K., Sattler, U., Drummond, N., Horridge, M., Rector, A.: Using owl to model biological knowledge. *International Journal of Human-Computer Studies* 65, 583–594 (2007)

19. Straccia, U.: Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research* 14, 137–166 (2001)
20. Yen, J.: Generalizing term subsumption languages to fuzzy logic. In: *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI 1991)*, pp. 472–477 (1991)
21. Zadeh, L.A.: Fuzzy sets. *Information and Control* 8, 338–353 (1965)
22. Zhao, J.: 1. *Advances in Semantic Computing*. In: *Uncertainty and Rule Extensions to Description Logics and Semantic Web Ontologies*. Technomathematics Research Foundation, pp. 1–22 (2010) (accepted)
23. Zhao, J., Boley, H.: *Canadian Semantic Web: Technologies and Applications*. In: *Knowledge Representation and Reasoning in Norm-Parameterized Fuzzy Description Logics*. Springer (2010)

Portfolio Investment Decision Support System Based on a Fuzzy Inference System

Isidoro J. Casanova

Department of Informatics and Systems, University of Murcia
Campus de Espinardo, 30100 Murcia, Spain

isidoroj@um.es

<http://webs.um.es/isidoroj>

Abstract. This paper describes a hybrid intelligent system formed by a decision support system based on rules for the management of a stock portfolio and by a fuzzy inference system to select the stocks to be incorporated.

This system simulates the behavior of any rational investor, so that each day would look if there is any investment opportunity with the use of technical indicators applying a fuzzy logic based approach.

The system has been tested in 3 time periods (1 year, 3 years and 5 years), simulating the purchase/sale of stocks in the Spanish continuous market and the results have been compared with the revaluations obtained by the best investment funds operating in Spain.

Keywords: Finance, Portfolio selection, Trading system, Decision support system, Fuzzy inference system.

1 Introduction

Investment management consists of *strategic asset allocation*, *tactical asset allocation*, and *stock picking* three phases [2]. *Strategic asset allocation* is a long-term allocation strategy that implies to elect the market where you are going to invest, what kind of assets you're going to buy and the distribution of these in the portfolio in accordance with the investor's objectives. *Tactical asset allocation* consists of regularly adjusting the portfolio, in a systematic or discretionary way, to take advantage of short-term opportunities. *Stock picking* consists in selecting the better stocks to be incorporated in the portfolio, being the most time-consuming phase and has a greater impact on the return of the portfolio.

Our study is focus on *tactical asset allocation* and *stock picking*.

To perform the *tactical asset allocation* an *intelligent system based on rules* that will dynamically invest in shares for a certain period of time is proposed.

The last phase, *stock picking*, is one of the most important phases, since it must decide for each day what shares should be bought to add to the portfolio. A simple rule that could be used to implement this step could be to invest in the stock that has been revalued at least a certain percentage in the last days. Although we could use more sophisticated rules using technical analysis, like Relative Strength Index (RSI) or Moving Average, [15].

Each of the indicators used in technical analysis has some limitation, and in most cases, the answer by each of them is not a definite “yes” or “no”. The best result could be achieved when combining many indicators at the same time and evaluating their output collectively.

Fuzzy reasoning is very effective in such environments. Managers perceive technical indicators differently and their answers to these indicators are not obvious and surely they are not true or false answers but somewhere in between. In fuzzy logic, the truth of any statement becomes a matter of degree. In our opinion, fuzzy logic blends very well with technical analysis process.

Our proposal is to use technical indicators with fuzzy logic in order to create a strict fuzzy indicator that only recommends “buy”, when most of the technical indicators recommend it. To implement this last phase we will use well defined and simple rules, selecting suitably input membership functions that influence in the purchase of a stock.

The proposed system will be applied to the Spanish stock market, in particular, the IBEX 35¹, keeping in mind that we invest all the available money in stocks without regard to which sector they belong, and supporting a maximum of 4 % loss per share. We will compare our investment performance with the index itself and with the results that have obtained the better equity funds that invest in the Spanish continuous market in the time limit of 1, 3 and 5 years.

In Section 2, we introduce the concepts of investment portfolio and how fuzzy inference systems have been introduced in order to help in financial market analysis. In Section 3, we present the system proposed with all the elements that compose it. In Section 4, we show different computational results that illustrate the behavior of the proposed hybrid intelligent system. Finally, we present the conclusions in Section 5.

2 Literature Review

2.1 Investment Portfolio

The concept behind investment portfolio is to combine different investment targets to avoid concentrating too much risk on any one target with the aim of dispersing overall investment risk. Any combination of two or more securities or assets can be termed an investment portfolio. Over a half century, the Markowitz meanvariance model, [13], has become a universally understood technique within the investment field. However, this model is limited by the uncertainty of the inputs such as expected returns, standard deviations, and correlation matrix. Many asset managers build on the foundation of the Markowitz meanvariance model to construct an Efficient Frontier portfolio and use the rate of return and variance to assess overall portfolio performance and risk. Nevertheless, this approach assumes that the rate of return and variance of each investment target is known at the beginning.

On the other hand *efficient-market hypothesis* is an idea partly developed in the 1960s by Eugene Fama and defended by Burton G. Malkiel [12] which asserts that financial markets are “informationally efficient”, or that prices on traded assets (e.g., stocks,

¹ The IBEX 35 (an acronym of Iberia Index) is the benchmark stock market index of the Bolsa de Madrid, Spain’s principal stock exchange.

bonds, or property) already reflect all known information, and instantly change to reflect new information. Therefore, according to theory, it is impossible to consistently outperform the market by using any information that the market already knows, except through luck.

Following this last hypothesis are the *passive managers*, which believe that it is impossible to predict which individual holdings or section of the market will perform better than another therefore their portfolio strategy is determined at outset of the portfolio and not varied thereafter. Many passive portfolios are index portfolio where the portfolio tries to mirror the market as a whole. Another example of passive management is the “buy and hold” method used by many traditional Unit Investment Trusts where the portfolio is fixed from outset. Today, there is a plethora of market indexes in the world, and thousands of different index funds tracking many of them.

One of the largest equity mutual funds, the Vanguard 500, is a passive management fund. The two firms with the largest amounts of money under management, Barclays Global Investors and State Street Corp., primarily engage in passive management strategies.

Contrary to the efficient-market hypothesis are the *active managers*, which believe that by selectively buying within a financial market that it is possible to outperform the market as a whole. Therefore they employ dynamic portfolio strategies buying and selling investments with changing market conditions.

Investors or mutual funds that do not aspire to create a return in excess of a benchmark index will often invest in an index fund that replicates as closely as possible the investment weighting and returns of that index; this is called *passive management*. *Active management* is the opposite of passive management, because in passive management the manager does not seek to outperform the benchmark index.

2.2 Artificial Intelligence and Investment Portfolio

The world of artificial intelligence applied to the stock market investment can be divided into two groups, the first one would try to ascertain as accurately as possible the future price of a share and the second endeavors in optimizing the mean-variance analysis proposed by Markowitz.

One of the most widely used techniques for predicting the price of a particular stock are neural networks, [20]. There are also used genetic algorithms to optimize the parameters of these neural networks or to optimize the technical analysis of one stock [7]. In [1] there are utilized, among others, as methods for the classification of stocks the recursive partitioning or probabilistic neural networks. There are also studies about how to predict the behavior of one stock from the news published on the Internet, [14].

With respect to Artificial Intelligence techniques to optimize the mean-variance analysis we have [11] that deals with fuzzy optimization schemes for managing a portfolio. Kendall and Su [9] use particle swarm optimization to find the best proportion of risk assets. This method is based on the mean-variance model and uses the Sharpe ratio as fitness function.

These articles may be helpful to invest, although they do not go so far as to realize the buying and selling of the stocks. Once stocks have been selected, the time of purchase and sale is not clearly specified, existing articles in which a notice appears in real-time

on what stock or group of stock are recommended, [18], where the stocks are bought and sold annually, [19], in which is estimated if a purchase is going to be long or short term, or where it just does not say anything about what methodology has been applied to estimate when these buying and selling must be realized.

In most of these articles there is a vector which indicates the percentage of investment being made in each stock for each one of the subperiods. For example, in [1] is fixed an investment horizon of 5 years, with sub-periods of 1 year. This modus operandi has not really considered the normal carry out of a trade operation done when an investor buys or sells shares. Thus, for example all these articles have a vector which shows the percentage of investment in each stock instead of real orders to buy or sell (market orders/limit orders). Likewise the commissions of buying and selling are not calculated in a completely real way and does not perform a daily monitoring of the stocks that are in the portfolio, so that these stocks could fall into a downward spiral and in none of these studies any corrective measure would be taken, either because until it does not pass the subperiod does not re-balance again the portfolio, or because the technique used to calculate the percentages of investment in each of the shares does not take into account the behavior of the stocks that have already been purchased.

2.3 Fuzzy Inference Systems

A fuzzy inference system is a computer paradigm based on fuzzy set theory, fuzzy if-then-rules and fuzzy reasoning.

Fuzzy inference systems have been successfully applied in fields such as automatic control, data classification, decision analysis, expert systems, and computer vision. Because of its multidisciplinary nature, fuzzy inference systems are associated with a number of names, such as fuzzy-rule-based systems, fuzzy expert systems, fuzzy modeling, fuzzy associative memory, fuzzy logic controllers, and simply (and ambiguously) fuzzy systems.

In the field of financial market analysis, we have for example, [6], which uses three technical indicators (rate of change, stochastic momentum and resistance indicator) in a fuzzy control system with the following modules: convergence (maps the technical indicators into new inputs), fuzzification, fuzzy processing and defuzzification (using the center of area method to map the output universe with four membership functions -low, medium, big and large- into a nonfuzzy action). Also in [5] the fuzzy trading system is based on four technical indicators (Moving Average Convergence/Divergence, Commodity Channel Index, Relative Strength Index and Bollinger Bands) and the output of the fuzzy system is a signal on a normalized domain, on which four different fuzzy sets (strong sell, sell, buy and strong buy) are defined.

On the other hand in [3] use a neuro-fuzzy based methodology to forecast the next day's trend of chosen stocks. The forecasting is based on the rate of change of three-day stock price moving average.

Furthermore, other scholars have used rough sets to generate trading rules. For instance, in [16] seven technical indicators are used for rule generation using three steps, namely, Decision Table Reconstruction, Discretization and Rough Sets Application. The last step is to apply rough sets to generate rules and check the predictive accuracy on the testing data sets.

3 Research Framework

3.1 Intelligent System for Tactical Asset Allocation

The proposed system for decision making is based on a policy of buying and selling the stocks that make up any stock market over a period of time.

This policy of buying and selling is based on that if we assume a stock market quoted from a start date ($date_{start}$) until an end date ($date_{end}$), each one of those days, all their stocks have had an opening price (p_{open}), a maximum price (p_{max}), a minimum price (p_{min}) and a closing price (p_{close}).

If we take a day d between $date_{start}$ and $date_{end}$ in which you have no shares purchased (there is no order of sale or purchase pending), then we will be able to select a set of m stocks (S_b), using technical analysis or any other technique, which would be most recommendable to buy, because it expects them to give a good return.

The technique for selecting stocks should calculate a value for each one of the stocks that make up the market on that day d , quantifying if it would be advisable to buy the shares. Stocks are ordered from most to least according to this value, and the system will have to choose the best set of stocks, defining which minimum value is considered for a stock to belong to this set of the better stocks, existing the possibility that one day any stock is not recommended ($m = 0$), or that many are recommended because its analysis has been the sufficiently satisfactory for all them.

Once we have this set S_b with the selection of the best stocks for a day d , we might try to buy all or some of these stocks. To simplify the algorithm we will try to buy only one of the selected stocks every day, so that after several days, we could have a portfolio of n stocks (S_a). Thus, to choose the stock to buy we would have two possibilities, to select the best or well to select any randomly.

Once we have chosen a stock, we will have to give the purchase order for the next day. We will limit the purchase price to any of the prices that has had the stock during that same day ($[p_{min}, p_{max}]$), or choose the opening p_{open} or closing p_{close} price of the next day. If we use a low purchase price, there are fewer possibilities to execute the purchase in the following days, but on the other hand, the stock will be bought more cheaply.

The next day ($d + 1$), one sees if the purchase of the share can be executed, whenever the purchase price of this day is between p_{min} and p_{max} . If this purchase order does not manage to be executed in W days, then we would eliminate this purchase order, and would give a new purchase order, selecting a share among the best ones of that day.

On this following day in which we can already have bought shares, it is necessary to calculate which of them should be kept in the portfolio and which should be sold, reason why a new analysis is performed to select p stocks (S_c) so that using again anyone of the previously commented techniques would tell us which stocks shall be maintained in the portfolio, since it is expected that they give a good profit value. The technique used must calculate a value for each of the stocks that make up the market on that day $d + 1$, quantifying if it would be advisable to maintain this share in the portfolio.

In this new day we would check if each one of the bought stocks continues being among the best that are recommended to maintain in the portfolio (S_c), in which case we would not do anything or otherwise we would leave spent M days without the stock

among the best to give a sale order. A sale order is given when M days have passed without the stock is recommended to keep in the portfolio or when the stock has had a loss regarding the price of purchase higher than a certain percentage $P\%$. A sale order will also have a limited price, as the purchase order. The higher the sale price more difficult it is to execute the sell order. In case during V days it is not possible to sell at this price, probably because the stock is in a bearish period, then we would descend the price of this sale order.

From this policy of buying and selling, it is proposed Algorithm II which simulate the intelligent behavior of an investor in a continuous market to form portfolios of n shares (S_a). In this algorithm the following functions are used:

- `exec_ord_sell_pend()`: Executes the pending sell orders that have been introduced in the system.
- `exec_ord_buy_pend()`: Executes the pending buy orders.
- `drop_price_ord_sell_pend_exec()`: Drops the sale price of the sale orders that carry without being executed during V days.
- `delete_ord_buy_pend_exec()`: Erases the purchase orders that they have not been executed in W days.
- `fuzzy_select_best_stocks_buy(date)`: Selects the best stocks that are recommended for purchase in a certain day. This function is explained thoroughly in the following section.
- `new_ord_buy(Best(S_b))`: Introduces in the system a new purchase order for the best share of S_b .
- `select_best_stock_hold (date, rule_hold_portf)`: Selects which shares are better to maintain in the portfolio, because it is expected that they provide a good profit value. The rule that realizes this selection "*rule_hold_portf*" is passed like input parameter to the algorithm.
- `new_ord_sell(A_i)`: Introduces in the system a new sale order for the share A_i , if this share surpasses the maximum loss permitted, or if the share is not recommended to maintain in the portfolio during more than M days.

3.2 Stock Picking Based on a Fuzzy Inference System

For every day d along the investment period it is necessary to look for which are the best shares S_b to be able to introduce them in the decision support system dedicated to tactical asset allocation commented in the previous paragraph, and that this one decides how it is going to invest in them.

To select the best shares of one day d , we are going to use technical analysis indicators, although the most difficult part of technical analysis is to decide which indicator to use.

We have chosen the following four technical indicators to select stocks:

1. Average Revaluation Period (ARP): Average revaluation that has had a stock in a given period of time.
2. Relative Strength Index (RSI): Relative Strength Index of a stock in a given period of time.

Algorithm 1. Tactical asset allocation

```

FUNCTION trade (datebegin, dateend, n, rule_hold_portf, moneybegin): moneyend
begin

    date=datebegin;
    money=moneybegin;
    while (date ≤ dateend)
    begin

        money=money+exec_ord_sell_pend();
        money=money-exec_ord_buy_pend();
        if (exists(Pending sell orders during  $V$  days))
            then drop_price_ord_sell_pend_exec();
        if (exists(Pending buy orders during  $W$  days))
            then delete_ord_buy_pend_exec();
         $S_b$ =fuzzy_select_best_stocks_buy(date);
        if ( $S_b < \{ \}$ ) and (money > 0) and ( $S_a < n$ )
            then new_ord_buy(Best( $S_b$ ));
         $S_c$ =select_best_stock_hold (date, rule_hold_portf);
        for each  $A_i$  in  $S_a$ 
            if ( $A_i$  not in  $S_c$  during  $M$  days) or (loss( $A_i$ ) >  $P$ )
                then new_ord_sell( $A_i$ );
        date=next_day(date);

    end;
    return money;

end;

```

3. Moving Average (MA): Calculates the revaluation that reaches a stock with respect to the average value of price in a given period of time.
4. Double Moving Average (DMA): Known also as double crossover method, uses a combination of long-term and short-term moving averages. When the shorter moving average rises above the longer moving average from below, a buy signal is issued.

The results of applying these indicators to the stocks are going to be the input variables in the fuzzy inference system. We have chosen these indicators because they are basic in the world of technical analysis, and because they are very easy to understand.

Technical analysis deals with probability and therefore multiple indicators can be used to improve the result. In most cases, the answer by each indicator is not a definite yes or no answer.

We are going to use technical indicators with fuzzy logic to create a strict fuzzy indicator that only recommends buying a stock when the set of indicators does it. We will only focus on the purchase recommendations, because the intelligent system discussed in the preceding section will be in charge of managing the portfolio, selling those shares no longer necessary.

Our plan can be summarized as follows:

- To create membership functions, where the inputs are each one of the financial indicators and the outputs are these indicators “fuzzified”.
- To create fuzzy rules that indicate if it is highly recommendable to buy a share.
- To translate the fuzzy output into a crisp trading recommendation.

Fuzzification. The input variables in this fuzzy inference system are mapped by sets of membership functions, known as “fuzzy sets”. The process of converting a crisp input value to a fuzzy real value between 0 and 1 is called “fuzzification”. The fuzzification comprises the process of transforming crisp values into grades of membership for linguistic terms of fuzzy sets. The membership function is used to associate a grade to each linguistic term.

Our fuzzy system also have a “ON-OFF” type of switch for the **Double Moving Average** input variable, because this input will always have a truth value equal to either 1 or 0, depending if the buy signal has been issued.

There is yet no fixed, unique, and universal rule or criterion for selecting a membership function for a particular “fuzzy subset” in general: a correct and good membership function is determined by the user based on his scientific knowledge, working experience, and actual need for the particular application in question.

The criteria followed in the fuzzification of profitability and RSI are explained below.

Fuzzification of Profitability. At the moment of carrying out the fuzzification of the daily profit values, the present fuzzy systems usually assign “low”, “normal” or “high” profitability values according to subjective estimations carried out by the writer of the article.

In this research, in order to perform this fuzzyfication we need to keep in mind that, statistically (Figure 1), is considered normal profitability between 0% and 0.5%-1%, high profitability between 0.5%-1% and 1.5%-4%, and very high profitability from 1.5%-4%, [4].

The membership grade functions defined on the profitability domain are based on trapezoid shapes (Figure 2). It can be seen that just positive returns have been considered, since only this type of profit values will be able to originate recommendations for purchase.

Fuzzification of RSI. The Relative Strength Index (RSI) method, which was developed by J. Welles Wilder, may be classified as a momentum oscillator, measuring the velocity and magnitude of directional price movements. Momentum is the rate of the rise or fall in price.

Wilder posited that when price moves up very rapidly, at some point it is considered overbought. Likewise, when price falls very rapidly, at some point it is considered oversold. In either case, Wilder felt a reaction or reversal is imminent. The slope of the RSI is directly proportional to the velocity of the move. The distance traveled by the RSI is proportional to the magnitude of the move.

As a result, Wilder believed that tops and bottoms are indicated when RSI goes above 70 or drops below 30. Traditionally, RSI readings greater than the 70 level are

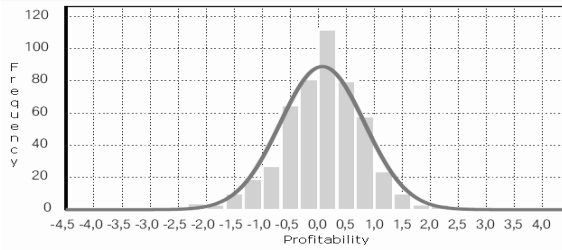


Fig. 1. Histogram of daily returns on IBEX35 (2003-2005)

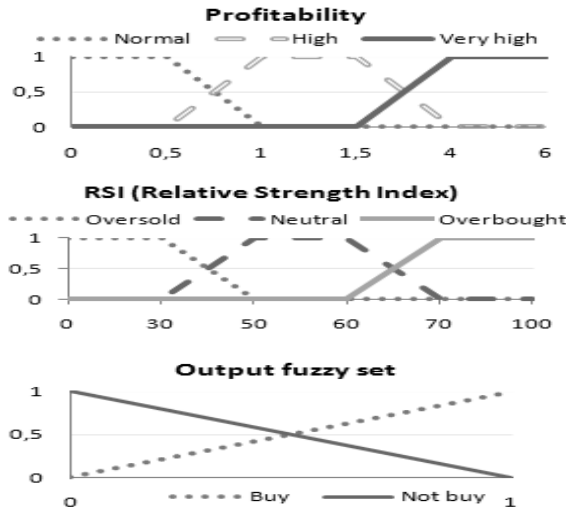


Fig. 2. Membership functions and output fuzzy sets

considered to be in overbought territory, and RSI readings lower than the 30 level are considered to be in oversold territory. In between the 30 and 70 level is considered neutral.

The membership grade functions defined on the RSI domain (Figure 2) have the following fuzzy sets (overbought, oversold and neutral) based on a trapezoid shape.

Fuzzy Rule Base. Decisions are made based on fuzzy rules. These rules are characterized by a collection of fuzzy IF THEN rules in which the preconditions and post-conditions involve linguistic variables. This collection of fuzzy rules characterizes the behavior of the system in a linguistic form that is close to the way human think.

Designing a good fuzzy logic rule base is key to obtaining a satisfactory controller for a particular application. Therefore, when designing the rules, it has been taken into account that only those rules that define the purchase of a stock must be defined, and they should be the easiest possible rules so as to understand its application. Thus, only the linguistic term associated with the most representative membership function to make

a purchase (“RSI oversold” or “profitability very_high”) has been used when setting these rules, and no other membership functions such as “profitability high” which are usually used in other researches.

For simplicity of design we have taken linear input-output relations (implications) in a SISO system. Generally, in multiple-input/multiple-output (MIMO) fuzzy inference systems, it is difficult to generate control rules.

Keeping the rules mentioned above in mind, the rules that we have defined are the following:

1. IF RSI is oversold THEN buy
2. IF DMA is buy_signal THEN buy
3. IF MA is very_high THEN buy
4. IF ARP is very_high THEN buy
5. OTHERWISE not_buy

The antecedent (the rules premise) describes to what degree the rule applies, while the conclusion (the rules consequent) assigns a membership function to the output variable.

The output variable “buy” is assigned a range between 0 and 1. A low value represents that is not a good idea to buy the stock and a high value represents an excellent opportunity to buy the stock. There is an inverse relationship between the output membership functions “buy” and “not_buy” so that: $\text{buy} = 1 - \text{not_buy}$.

The strength of the i th fuzzy rule is calculated by evaluating the strength of the precondition i (degree of truth) on the corresponding output membership. The final value of the output variable will correspond exactly with the value that reaches the membership function in the precondition.

Combining Rules and Defuzzification. As all the rules are activated every day resulting in different values for the output fuzzy set “buy”, corresponding each output value with the value of the fuzzified input, we are going to perform a combination of rules additive, [10], to obtain a unique final value assigning a weight to each rule.

The weight of the combiner can be thought of as providing degrees of belief to each rule, but we consider that all the rules have the same importance, so we set all the weights equal to unity.

Defuzzification is a mapping process from a fuzzy space defined over an output universe of discourse into a nonfuzzy (crisp) action. It is not a unique operation as different approaches are possible.

The final output of the system (a crisp control signal) is a value between 0 and 1. A strong buy signal is generated when the output is close to 1.0 and a strong not buy signal is generated when the output is close to zero.

The fuzzy logic and the fuzzy control rules are considered and are chosen so that the defuzzified output is always a linear function of the inputs to the fuzzy controller. According to [17] the output of multiple input single output fuzzy logic controller can be represented by the convex linear combination of the inputs of fuzzy logic controller.

Therefore, to calculate the final output of the system we calculate the average of the fuzzified values that have been returned by the selected membership functions.

4 Experiments and Results

We are going to verify the operation of the system in 3 periods of time: 5 years (2005-2009), 3 years (2007-2009) and 1 year (2009). The market where we will operate will be the Spanish stock market, but restricted to the shares that conformed the IBEX35 in the year 2009. The historic prices have been corrected of dividends, splits and increases in capital. The short selling is not allowed. The cost of each trade has been taken into consideration, so that we assume that the financial intermediary charges a fee of 0.2% and we are going to consider the transaction fees published by the market of Madrid.

The portfolio will be formed by 14 shares as maximum and the rule which is responsible for defining whether a stock should remain in the portfolio has been defined so that the Relative Strength Index of the shares for 28 days must be worth at least 45 for not give a sale order if the stock has remained in the portfolio at least 14 days. The maximum loss allowed to give an immediate sale order is 4%.

In Table 1 are the results obtained in each one of the three periods and they are compared with the revaluation of IBEX 35 in that time. Also is the variance of the daily revaluation throughout each one of the periods. We can found that the system achieves a lower variance than produced by the IBEX 35 and therefore offers less risk. This is because the intelligent system for tactical asset allocation controls the behavior of the shares, selling for example those that have lost over 4%.

Table 1. Result of the simulation vs. IBEX35 revaluation

System	Period	Result	Variance
Simulation	2009	42.94%	1.65
IBEX35	2009	26.22%	2.45
Simulation	2007-2009	5.22%	1.42
IBEX35	2007-2009	-16.92%	3.34
Simulation	2005-2009	104.23%	1.12
IBEX35	2005-2009	30.85%	2.23

Figure 3 shows graphically a comparison of daily results of the hybrid intelligent system with the behavior of the IBEX35 index in the period 2005-2009, with an initial investment of 100,000 Euros.

To evaluate the importance of the results obtained with this system, we are going to compare them with the results of a report elaborated by INVERCO (Spanish Association of Investment and Pension Funds). Table 2 shows the ranking (R) by annual equivalent return (APR) in periods of 1, 3 and 5 years of each one of the best Spanish equity funds (Foncaixa Bolsa España 150, BBVA Bolsa Ibex Quant, Bankinter Bolsa España 2, CC Borsa 11, Venture Bol. Española), until 31 December 2009.

In this report elaborated by INVERCO we can see that the most profitable fund in 2009 was the *Foncaixa Bolsa España 150*, with a 51.9% of revaluation, although this fund was ranked in the position 79 by the return of -8.9% APR that obtained in the 3 previous years. Our system obtains in 2009 a yield of 42.9%, so that if we could participate in this ranking we would be included in the fifth position by yield to one year.



Fig. 3. Daily evolution of the investment and the IBEX35

Table 2. Ranking of funds from Spanish equity investment [8]

	2009		2007-2009		2005-2009	
	APR	R	APR	R	APR	R
Foncaixa	51.9	1	-8.9	79	-	-
BBVA	48.7	2	-13.0	86	-	-
Bankinter	34.5	32	4.4	1	11.7	1
CC	19.1	88	2.4	2	4.1	72
Venture	34.9	24	0.0	6	10.6	2
Simulation	42.9	5	1.7	3	15.4	1

To three years view the stock market crisis continues nevertheless passing bill, since almost all the funds register red numbers, except the first funds, like *Bankinter Bolsa España 2*, that with a 4.4% APR would remain first inside this ranking of the better investment funds in Spain for 3 years. Our system achieves a revaluation of 1.7% APR in this period, therefore we would remain third in the ranking for profitability for 3 years.

With a horizon of five years, the situation is different: some funds, as the mentioned *Bankinter Bolsa España 2* (11.7% APR) or *Venture Bol. Española* (10.6% APR) obtain notable performances, although our system surpasses all of these funds with a profit value of 15.4% APR.

5 Conclusions

This article has proposed a hybrid intelligent system that solves quite successful investment in shares forming a portfolio. This system has two main parts: the first is responsible for buying and selling shares, managing a portfolio and monitoring the purchased shares and the second part is responsible for selecting which are the best shares to incorporate them into the portfolio.

The part entrusted to realize the tactical asset allocation, corresponds to a decision system based on rules and the part entrusted to select shares has been based on a fuzzy inference system.

The proposed system solves in an integral way the process of investment in shares, since usually all the research papers focus on stock selection leaving out portfolio management, not taking into account the way in which an investor operates normally when he carries out the purchase or sale of shares. Moreover, the implementation of stock picking is also novel, using only purchase rules and properly selecting the membership functions that have more influence in the purchase of a stock.

In the obtained results the revaluation of the reference index is surpassed (IBEX35) in all the periods and even we can place the hybrid intelligent system in the first positions of the ranking by profit value if it is compared with commercial investment funds that invest in Spanish equities.

Acknowledgements. Supported by the project TIN2008-06872-C04-03 of the MICINN of Spain and European Fund for Regional Development.

References

1. Albanis, G., Batchelor, R.: Combining heterogenous classifiers for stock selection. *International Journal of Intelligent Systems in Accounting and Finance Management* 15(1-2), 1–21 (2007)
2. Amenc, N., Sourd, V.L.: *Portfolio Theory and Performance Analysis*. Wiley Finance (2003)
3. Atsalakis, G., Valavanis, K.: Forecasting stock market short-term trends using a neuro-fuzzy based methodology. *Expert Systems with Applications* 36(7), 10696–10707 (2009)
4. BME - Bolsas y Mercados Españoles: Normalidad de las series financieras (2009), <http://www.bolsasymercados.es>
5. Cheung, W., Kaymak, U.: A fuzzy logic based trading system. In: *3rd European Symposium on Nature-inspired Smart Information Systems*. Nisis (2007)
6. Dourra, H., Siy, P.: Investment using technical analysis and fuzzy logic. *Fuzzy Sets and Systems* 127(2), 221–240 (2002)
7. Drake, A., Marks, R.: *Genetic Algorithms in Economics and Finance: Forecasting Stock Market Prices and Foreign Exchange - A Review*. Genetic algorithms and genetic programming in computational finance. Kluwer (2002)
8. INVERCO - Asociación de Instituciones de Inversión Colectiva y Fondos de Pensiones: Ranking Fondos Inversión a (December 31, 2009), <http://www.inverco.es>
9. Kendall, G., Su, Y.: A particle swarm optimisation approach in the construction of optimal risky portfolios. In: *23rd IASTED International Multi-conference Artificial Intelligence and Applications*, pp. 140–145 (2005)
10. Kosko, B.: *Neural network and Fuzzy systems*. In: *A Dynamical Systems Approach to Machine Intelligence*. Prentice Hall Press (1992)
11. León, T., Vercher, V.L., Viability, E.: Viability of infeasible portfolio selection problems: A fuzzy approach. *European Journal of Operational Research* 139(1), 178–189 (2002)
12. Malkiel, B.: *A Random Walk Down Wall Street*. W. W. Norton & Company (1973)
13. Markowitz, H.: Portfolio selection. *Journal of Finance* 7(1), 77–91 (1952)
14. Mittermayer, M.: Forecasting intraday stock price trends with text mining techniques. In: *37th Annual Hawaii International Conference on System Sciences*, pp. 140–145. IEEE Xplore (2004)
15. Murphy, J.: *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. Prentice Hall Press (1999)

16. Shen, L., Loh, H.T.: Applying rough sets to market timing decisions. *Decision Support Systems* 37(4), 583–597 (2004)
17. Sun, H., Liu, L.: A linear output structure for fuzzy logic controllers. *Fuzzy Sets and Systems* 131(2), 265–270 (2002)
18. Tseng, C.-C., Gmytrasiewicz, P.J.: Real time decision support system for portfolio management. In: 35th Annual Hawaii International Conference on System Sciences, vol. 3, p. 79. IEEE Computer Society (2002)
19. Zargham, M., Sayeh, M.: A web-based information system for stock selection and evaluation. In: Proc. International Conference on Advance Issues of E-Commerce and Web-Based Information Systems, pp. 81–83. IEEE Computer Society (1999)
20. Zekic, M.: Neural network applications in stock market - a methodology analysis. In: Proc. 9th International Conference on Information and Intelligent Systems, pp. 255–263 (1998)

Fuzzy Analytical Network Models for Metasearch

Arijit De¹ and Elizaebeth Diaz²

¹ TCS Innovation Labs-Mumbai, Tata Consultancy Services,
Thane (W), Mumbai 400601, India

² University of Texas of Permian Basin, Odessa, TX 79762, U.S.A.
arijit.axd9142@gmail.com, diaz_e@utpb.edu

Abstract. Merging of search engine results is a key metasearch engine function. Most result merging models try to merge ranked lists of web documents returned by search engines in response to a user query using some linear combination approach. A few give more importance to one search engines as opposed to another based on some performance criteria. Other assign weights to documents ranks etc. However few models compare documents and search engines head to head during the process of result merging. In this paper we propose two models for result merging for metasearch, Fuzzy ANP and Weighted Fuzzy ANP that employ fuzzy linguistic quantifier guided approach to result merging using Saty's Analytical Network Process. We compare our models to existing result merging models. Our results show significant improvements.

Keywords: Information retrieval, Fuzzy sets, Soft computing, Multi-criteria decision making.

1 Introduction

A metasearch engine is an Information Retrieval (IR) system that employs multiple search engines, in parallel to search for information on the World Wide Web (WWW). Search engines return documents/web pages relevant to the query as a ranked result list of documents. The metasearch engine then merges these lists into one list for the user. Metasearch engines serve to expand the scope of web search beyond the coverage of one search engine.

Result merging is thus a critical problem for metasearch. In this paper we take a multi criteria decision making (MCDM) approach to the result merging problem by proposing two models for result merging, Fuzzy ANP and Weighted Fuzzy ANP. Our models are based on the Analytical Network Process (ANP), a MCDM technique proposed by Saty [12] and use Fuzzy Linguistic Quantifiers proposed by Zadeh [16]. We compare the performance of our models with the OWA model [4], the Borda-Fuse and Weighted Borda Fuse models[1].

This paper is organized as follows. In the next section we do a brief review of the existing result merging models with a special focus on the OWA model for result merging proposed by Diaz [4]. In subsequent sections we discuss our proposed Fuzzy ANP and Weighted Fuzzy ANP models. Following this we describe our experiments, take a look at our results and summarize our discussions in a conclusion.

2 Previous Work

Early result merging models, such as the Logistic Regression [7] model and the Combination of Experts [13] model combined document scores/ranks through a linear combination function. However, the most popular linear combination model was the Borda-Fuse model and the Weighted Borda Fuse models proposed by Aslam and Montague [1]. Diaz [5] applied Yager's [14] OWA operator to create a result aggregation model for metasearch. As we compare our work to the OWA model, the Borda-Fuse model, and the Weighted Borda-Fuse models we discuss these in greater depth.

2.1 Borda-Fuse and Weighted Borda-Fuse Models

The Borda-Fuse model was proposed by Aslam and Montague [1]. It is based on the Borda-Count [3]. In this model the highest ranked document in each result list returned by a search engine is given a specific number of "Borda" points. Let us say d points are assigned to the top document. The next document receives $d-1$ Borda points and so on. If there are some documents that exist in some result lists but are missing in others, these are assigned a less number of points in the lists they do not appear in. The documents are ranked in descending order according to the total number of points accumulated in these lists. The Weighted Borda-Fuse is a weighted aggregation model that ranks documents based on the product sum of the Borda points earned by a document from a search engine and an importance weight attached to that search engine.

2.2 OWA Model for Metasearch

Diaz [4] applies the OWA operator for result aggregation in a metasearch model. The OWA model uses a measure similar to Borda points, called positional values. The positional value (PV) of a document d_i in the result list l_k returned by a search engine s_k is defined as $(n - r_{ik} + 1)$ where, r_{ik} is the rank of d_i in search engine s_k and n is the total number of documents in the result. Thus, the top ranked document in a result list has the highest positional value. One shortcoming of the Borda-Fuse model is that it handles missing documents (ones that appear in some lists to be merged but not in others) in a rather ambiguous way by distributing the remaining points available to them. This often results in missing documents being ranked at the bottom of the list. The issue of missing documents has been discussed by Meng [9] where the authors mention the challenge posed by them to result merging. Reasons for missing documents are obvious as coverage of search systems vary.

Diaz [4] addresses this issue by applying two simple heuristics for handling missing documents by calculating a virtual positional value of a document missing in the result list depending on its positional value in lists they occur, thereby effectively inserting the document in the result list in which it is missing. Handling missing documents makes result merged a easier task.

Yager [14] proposed the OWA operator as multi-criteria decision making (MCDM) approach. Let A_1, A_2, \dots, A_n be n criteria of concern in a multi-criteria

decision making problem and x be a alternative, being rated by/against these criteria. $A_j(x) \in [0, 1]$ indicates the degree to which x satisfies the j^{th} criteria. Yager [14] comes up with a decision function F to combine these criteria and evaluate the degree to which the alternative x satisfies the criteria. Let $a_1=A_1(x)$, $a_2=A_2(x)$, and $a_n=A_n(x)$. The OWA decision function is:

$$F(a_1, a_2, \dots, a_n) = \sum_{i=1}^n w_i \bullet b_i \tag{1}$$

Here b_j is the j^{th} greatest a_i . Here w_j is the ordered weight vector attached to the j^{th} criteria and such that the ordered weight vector $W = [w_1, w_2, \dots, w_n]$ associated with the OWA operator is key to determining the ‘‘orness’’ of the aggregation. When $W = [1, 0, \dots, 0]$ the value of the largest a_i is the value of F . In this case we get maximum orness. On the other hand when $W = [0, 0, \dots, 1]$ the value of the smallest a_i is the value of F . In this case we get minimum orness/maximum andness.

In the OWA model for metasearch, Diaz [5] uses the Yager [15] approach to computing OWA weights using linguistic quantifiers. The weight associated with the i^{th} criterion (positional value associated with a search engine) is given by:

$$w_i = Q\left(\frac{i}{n}\right) - Q\left(\frac{i-1}{n}\right) \tag{2}$$

Here, Q is a RIM (Regular Increasing Monotone) quantifier of the form $Q(r) = r^\alpha$. The orness associated with the quantifier, orness (Q) = $\frac{1}{1 + \alpha}$. In the OWA model,

each result list, returned by a search engine, is analogous to a criterion and each document an alternative. The positional value of the document in a search engine result list corresponds to the extent to which a document (alternative) satisfies a search engine (criteria) for a specific query. It is now easy to apply the OWA operator as a decision function to compute the value of function F , for each document. Sorting the documents in descending order depending on the value of function F gives us the merged list of documents.

2.3 Shortcomings of the OWA Model

The OWA model for metasearch assigns weights to the positional values of documents based on the order. While it is comprehensive in handling missing documents, it does not explore the relationship between documents and search engines in pair-wise comparisons. Saty [11] highlights the advantages of pair-wise comparisons in MCDM problems. To create a model that explores the relationship between documents and search engines, we came with the Fuzzy ANP model for metasearch. We further extended this model to assign search engine importance weights in the Weighted Fuzzy ANP model.

3 Proposed Model

In the last part of the previous section we discussed the shortcomings of the OWA model. Our main motivation was to overcome these and build a model that analyzed

the close relationship between documents and search engines in pair-wise comparison. While Saty's Analytical Hierarchy Process (AHP) is a more popular MCDM approach, we chose to build our model on the more generic Analytical Network Process (ANP) as the core structure of the metasearch problem is not hierarchical in nature. In this section we discuss ANP, without going into the mathematical theory behind its creation. Then we proceed to describe how Fuzzy Linguistic Quantifiers developed by Yager [15] is used in transforming the ANP Super Matrix to a Weighted (column stochastic) Super Matrix.

3.1 Analytical Network Process

Saty proposed two MCDM techniques, the Analytical Hierarchy Process (AHP) [10] and the Analytical Network Process (ANP) [12], to solve MCDM problems that compared criteria and alternatives in pairs. While the AHP is considered the technique of choice for most MCDM problems, the ANP is used when the problem cannot be structured hierarchically because the problem involves the interaction and dependence of higher level elements on a lower level element [12]. Moreover, when the problem is not hierarchical in nature the Analytical Network Process (ANP) is more appropriate.

The first step in the ANP process is model construction and problem structuring. In this step the key components in the model, alternatives and criteria need to be clearly identified and their relationships captured through the creation of a network. The structure can be obtained by the opinion of decision makers through brainstorming or other appropriate methods.

The second step is the creation of pair-wise comparison matrices and priority vectors. In ANP decision elements at each component are compared pair-wise with respect to their importance towards their control criterion, and the components themselves are also compared pair-wise with respect to their contribution to the goal. Pair-wise comparisons of two alternatives or two criteria at a time can be done quantitatively or by discussing with experts. In addition, if there are interdependencies among elements of a component, pair-wise comparisons also need to be created, and an eigenvector can be obtained for each element to show the influence of other elements on it. The relative importance values are determined with Saaty's 1-9 scale (Table 1), where a score of 1 represents equal importance between the two elements and a score of 9 indicates the extreme importance of one element (row component in the matrix) compared to the other one (column component in the matrix).

Let us formalize the notion of pair-wise comparisons and construction of the super matrix. Let us say we have a set of alternatives $A = \{a_1, \dots, a_p\}$ and a set of criterion $C = \{c_1, \dots, c_q\}$. Using the 9 point scale shown in table 1, we can compare alternatives pair-wise for each criterion, based on the degree to which the alternative satisfies the criterion. Thus for each alternative c_i in C we can obtain a pair-wise matrix M . Each element of the matrix M , m_{jk} represents a quantified result of pair-wise comparison of alternatives a_j and a_k . Here $1 \leq m_{jk} \leq 9$ as per the 9 point scale shown in table 1. To obtain the priority vectors we divide each element of the matrix M by the sum of the column and then average out the values. Thus we can obtain for each criteria c_i , a priority vector $V = \{V_j, \text{ where } 1 \leq j \leq p\}$ and each V_i represents the alternative a_j . Thus for each (c_i, a_j) we get a value V_{ij} . Similarly, criteria can also be

compared pair-wise with reference to alternatives, depending on how each pair of criteria (c_i, c_j) measure up with respect to an alternative, for all c_i, c_j in C . Similarly priority vectors can be created for each alternative a_k such that we obtain a priority value V_{ki} for (a_k, c_i) .

The third step in the process is to create a super matrix. The super matrix concept is similar to the Markov chain process. To obtain global priorities in a system with interdependent influences, the local priority vectors are entered in the appropriate columns of a matrix. As a result, a super matrix is actually a partitioned matrix, where each matrix segment represents a relationship between two nodes (components or clusters) in a system. To put it simply the super matrix is a matrix that contains each priority vector corresponding to criteria and alternatives. The super matrix is a square matrix with each alternative and each criteria being a row element and as well as a column element. Each priority vector for an alternative and criterion is placed in the column for that alternative or criterion in the super matrix.

The super matrix created must be raised to a higher power till it converges to a limiting super matrix. Convergence occurs when each column of the super matrix contain identical values. Thus final scores are obtained for each alternative from their corresponding row values in the limiting super matrix. However for the initial super matrix created to converge it needs to be column stochastic. This means that all column values need sum up to 0. Thus prior to creating a limiting super matrix, each element in every column of the super matrix needs to weighted such the sum of elements in the column need to sum up to unity. This intermediate step results in the creation of a weighted super matrix.

Table 1. Pair-wise comparison matrix scale

m_{jk}	Criterion Value
1	If a_i is equally important as a_j
3	If a_i is weakly more important than a_j
5	If a_i is strongly more important than a_j
7	If a_i is very strongly more important than a_j
9	If a_i is absolutely more important than a_j
1/3	If a_i is weakly less important than a_j
1/5	If a_i is strongly less important than a_j
1/7	If a_i is very strongly less important than a_j
1/9	If a_i is absolutely less important than a_j

3.2 Linguistic Quantifiers

While the backbone of our proposed models, Fuzzy ANP and the Weighted Fuzzy ANP, is the Analytical Network Process, we use a Fuzzy Linguistic Quantifier Guided approach to transforming the Super Matrix constructed through pair-wise comparisons, into a column stochastic Weighted Super-Matrix. Let us introduce the concept of Fuzzy Linguistic Quantifiers by reviewing the work of Zadeh [16] and Yager [15] in brief.

Linguistic quantifiers have been used to generate ordered weights for aggregation in the OWA operator in Yager [14, 15]. The notion of linguistic quantifiers was introduced by Zadeh [16]. Linguistic quantifiers are the mathematical representation

of linguistic terms such as *at most*, *many*, *at least half*, *some* and *few*. In Zadeh [16] suggests a formal representation of these linguistic quantifiers using fuzzy sets. In classical logic, only two fundamental quantifiers are used. These quantifiers are “there exists” a certain number and “all.” Furthermore, Zadeh breaks up quantifiers into two types: absolute and relative. Absolute quantifiers can be represented as zero or positive real numbers, such as “about 5,” “greater than 10.” Relative quantifiers are terms such as “most,” “few,” or “about half.” Yager [15] distinguished three categories of these relative quantifiers. Of these the most popular quantifier is the Regular Increasing Monotone (RIM) quantifier of the form $Q(r) = r^\alpha$, mentioned earlier. Yager [15] shows how to model these quantifiers, to obtain weights for his OWA operator. Equation 2 shows a technique to generate weights when criteria importances are not taken into consideration. However when criteria/alternative importances are available Yager uses equation 3 to compute weights.

$$w_j(x) = Q\left(\frac{\sum_{k=1}^j u_k}{T}\right) - Q\left(\frac{\sum_{k=1}^{j-1} u_k}{T}\right) \tag{3}$$

Here u_k is the importance weight of the k^{th} criteria to be merged. One property of the weights so generated is that they always add up to unity. We exploit this in the construction of the Weighted Super Matrix.

In our proposed models for metasearch we borrow this notion of linguistic quantifier guided weights in transforming the constructed Super Matrix to the Weighted (column stochastic) Super Matrix. Let us illustrate the working with the help of an example. Let us say that a column of our super matrix constructed is of the form $[0, 0, 0, 0.8, 0.6, 0.4]^T$. Clearly these values do not add up to unity and therefore the column is not stochastic. To transform this column into a column stochastic matrix we compute Fuzzy Linguistic Weights using the equation 3. Here u_1, u_2 and u_3 are 0 while $u_4 = 0.8, u_5 = 0.6$ and $u_6 = 0.4$. Let us say we apply a weight of $\alpha = 1$ (for simplicity). Weights w_1, w_2 and w_3 are 0. Weight $w_4 = 0.44, w_5 = 0.337$ and $w_6 = 0.222$. Now our column becomes $[0, 0, 0, 0.44, 0.337, 0.222]$.

3.3 The Fuzzy ANP Model

Our proposed model Fuzzy ANP is based on Saty’s [12] Analytical Network Process (ANP). In our model, in order to apply the Analytical Network Process, we treat our search engines (criteria) and documents (alternative) as nodes in a network. The steps are outlined below.

Step 1: Modeling Documents and Search Engines Relationships. Let us say we are trying to merge result lists from n search engines $SE\text{-}SET = \{ SE_1, SE_2, \dots SE_n \}$. Let $DOC\text{-}SET$ be a set of m unique documents returned in all. Thus $DOC\text{-}SET = \{ D_1, D_2, \dots D_m \}$. Each document and search engine appears as nodes in an Analytical Network $G(E, V)$. Here E represents a set of edges and V represents a set of vertices. Since each search engine and document is a vertex, $|V| = m+n$ and $V = \{ SE_1, SE_2, \dots SE_n, D_1, D_2, \dots D_m \}$. $E = \{ E_{ik}, 1 \leq i \leq n \ \& \ 1 \leq k \leq m \}$. E_{ik} is the rank/score assigned by search engine SE_i to document D_k . $E_{ik} = 0$ when the document D_k is not retrieved by search engine SE_i .

Step 2: Pair-wise Comparisons of Documents and Search Engines. Let us compare documents pair-wise. Let us say a search engine $SE_f \in SE\text{-}SET$ return relevance scores, E_{fi} and E_{fk} for two documents D_i and D_k respectively. $D_i, D_k \in DOC\text{-}SET$. Let $E_{f,max}$ and $E_{f,min}$ be the maximum and minimum scores returned by a search engine SE_f for any document. The pair-wise comparison value $P(SE_f, D_i, D_k)$ for all $D_i > D_k$ is shown in equation 4. Consequently as per Saty's scale $P(SE_f, D_k, D_i) = 1/P(SE_f, D_i, D_k)$.

$$P(SE_f, D_i, D_k) = \frac{E_{fi} - E_{fk}}{E_{f,max} - E_{f,min}} \bullet 9 \tag{4}$$

Equation 4 holds when $E_{f,max} \neq E_{f,min}$ and $E_{fi} \neq E_{fk}$ and. If $E_{f,max} = E_{f,min}$ or $SC_i = SC_k$ then $P(SE_f, D_i, D_k) = 1$. The pair-wise comparison values $P(SE_f, D_i, D_k)$ can be computed for all $D_i, D_k \in DOC\text{-}SET$ and stored in a matrix, which can be normalized by dividing each column by a sum of all elements in the column and then by taking the average of each row to obtain a priority vector $VSE_f = [SD_{f1}, SD_{f2}, .. SD_{fm}]$. Here $SD_{f1}, SD_{f2}, .. SD_{fm}$ are pair-wise comparison scores for documents $D_1, D_2, \dots D_m$ with respect to SE_f .

Similarly let us say we have a document $D_f \in DOC\text{-}SET$. Two search engines SE_i and SE_k return relevance scores of E_{if} and E_{kf} for D_f . Here, $SE_i, SE_k \in SE\text{-}SET$. Let $E_{max,f}$ and $E_{min,f}$ be the maximum and minimum scores obtained by D_f . $P(D_f, SE_i, SE_k)$ can be computed for all $SE_i, SE_k \in SE\text{-}SET$ using equation (5) and stored in a matrix.

$$P(D_f, SE_i, SE_k) = \frac{E_{if} - E_{kf}}{E_{max,f} - E_{min,f}} \bullet 9 \tag{5}$$

This matrix can be normalized, to get the priority vector, $VD_f = [SSE_{f1}, SSE_{f2}, .. SSE_{fn}]$ specific to document D_f . Here $SSE_{f1}, SSE_{f2}, .. SSE_{fn}$ are scores returned for $SE_1, SE_2, \dots SE_n$ as a result of pair-wise comparison.

Step 3: Constructing the Super Matrix. Let SM be the super-matrix where each document and search engine is represented by a row and a column. Thus SM is of square matrix of dimension $lm+nl$. For each document D_f , from Step 2, we have a priority vector $VD_f = [SSE_{f1}, SSE_{f2}, .. SSE_{fn}]$ representing search engines $SE_1, SE_2, .. SE_n$. We populate the super matrix SM such that $SM(SE_1, D_f) = SSE_{f1}, SM(SE_2, D_f) = SSE_{f2}, \dots, SM(SE_n, D_f) = SSE_{fn}$. For each search engine SE_f we have a vector $VSE_f = [SD_{f1}, SD_{f2}, .. SD_{fm}]$ representing documents $D_1, D_2, .. D_m$. We populate the super matrix SM such that $SM(D_1, SE_f) = SD_{f1}, SM(D_2, SE_f) = SD_{f2}, \dots, SM(D_m, SE_f) = SD_{fm}$.

Step 4: Transforming the Super Matrix to form a Weighted Super Matrix. For the Super Matrix to converge we need to transform it to a column stochastic Weighted Super Matrix. This is done by applying weights to elements in each column such that all column values add up to unity. We take the column values and use them as inputs in computing linguistic fuzzy weights as developed by Yager [15] and described in equation 3 and the subsequent example (section 3.2). This makes the matrix column stochastic as the linguistic fuzzy weights add up to unity. The column stochastic Weighted Super Matrix is then multiplied with itself iteratively till the values of each column becomes identical. The column value corresponding to the each document determines the final score of the document.

Example

Let us illustrate the working of our model with an example. Let us consider a metasearch system where four documents D1, D2, D3 and D4 are being ranked. Let us say there are 3 search engines whose results are to be merged SE1, SE2 and SE3. The scores achieved by each document from each search engine are represented below in table 2. Documents can be sorted by their scores obtained from each search engine to form ranked result lists. Search Engine SE2 does not retrieve document D1. Hence the corresponding scores are 0.

Table 2. Document Scores for Search Engines

	SE1	SE2	SE3
D1	4.14	0.00	2.04
D2	5.78	4.81	4.71
D3	3.18	3.69	3.62
D4	2.20	1.98	2.94

The first step is to construct a network graph with each node representing a document or a search engine. Each document node (represented as an oval) is connected to a search engine node if the search engine retrieves the document. Notice that there is no edge connecting SE2 with D1 as SE2 does not retrieve document D1.

From the Table 2, using equation 4, we can do pair-wise comparison of documents for each search engine. Table 3 shows the results of pair-wise comparison for documents using search engine SE2. Each column in Table 4 represents a document priority vectors using a particular search engine. The column headers contain the names of the search engines. Column marked SE2 from Table 4 represents priority vector for documents D1 through D4 using SE2. The priority vector values are obtained by dividing each element of the matrix in Table 3 by the sum of each column and then computing the average of each row. Notice that since SE2 does not retrieve documents D1 there is no pair-wise comparison for other documents with D1 for this search engine.

Table 3. Document Pair-Wise comparisons using SE2

SE2	D1	D2	D3	D4
D1	0.00	0.00	0.00	0.00
D2	0.00	1.00	2.11	5.29
D3	0.00	0.47	1.00	3.18
D4	0.00	0.19	0.31	1.00

Similarly, we can calculate Priority Vectors for Search Engine SE1, SE3, and SE4 for each document and represented in columns in table 5.

Using these priority vectors we construct the super matrix. The super matrix is shown in Table 6. Notice that the super matrix is not column stochastic. This is where Fuzzy Linguistic Weights are employed to calculate the weighted super matrix that is column stochastic.

Table 4. Priority Vectors of Documents using Search Engines for Pair-Wise Comparisons

	SE1	SE2	SE3
D1	0.21	0.00	0.05
D2	0.63	0.44	0.61
D3	0.10	0.23	0.22
D4	0.05	0.08	0.12

Table 5. Priority Vectors of Search Engines using Documents for Pair-Wise Comparisons

	D1	D2	D3	D4
SE1	0.55	0.81	0.06	0.13
SE2	0.00	0.09	0.50	0.08
SE3	0.12	0.10	0.44	0.79

Table 6. Super Matrix

	SE1	SE2	SE3	D1	D2	D3	D4
SE1	0.00	0.00	0.00	0.55	0.81	0.06	0.13
SE2	0.00	0.00	0.00	0.00	0.09	0.50	0.08
SE3	0.00	0.00	0.00	0.12	0.10	0.44	0.79
D1	0.21	0.00	0.05	0.00	0.00	0.00	0.00
D2	0.63	0.44	0.61	0.00	0.00	0.00	0.00
D3	0.10	0.23	0.22	0.00	0.00	0.00	0.00
D4	0.05	0.08	0.12	0.00	0.00	0.00	0.00

Let us consider the column for SE2. The column values are [0, 0, 0, 0, 0.44, 0.23, 0.08]. We treat the values in the column as weights. Thus $u_1=0, u_2=0, u_3=0, u_4=0, u_5=0.44, u_6=0.23, u_7=0.08$. Applying equation 4, we get $T = 0.75$. Using a linguistic quantifier of the form $Q(r) = r^\alpha$ where $\alpha = 2$ we obtain the weights $w_1=0, w_2=0, w_3=0, w_4=0, w_5=0.35, w_6=0.45, w_7=0.20$. These values sum up to unity and therefore the column is column stochastic. Similar adjustments can be made for other columns. Table 7 shows the weighted super matrix.

Table 7. Weighted Super Matrix

	SE1	SE2	SE3	D1	D2	D3	D4
SE1	0.00	0.00	0.00	0.67	0.66	0.00	0.02
SE2	0.00	0.00	0.00	0.00	0.16	0.31	0.03
SE3	0.00	0.00	0.00	0.33	0.18	0.69	0.95
D1	0.04	0.00	0.00	0.00	0.00	0.00	0.00
D2	0.67	0.35	0.43	0.00	0.00	0.00	0.00
D3	0.19	0.45	0.35	0.00	0.00	0.00	0.00
D4	0.10	0.20	0.22	0.00	0.00	0.00	0.00

Multiplying this square matrix by itself iteratively results in the columns becoming identical in content. This matrix where the columns become identical in content is called the limiting super matrix. Table 8 shows the limiting super matrix. Document scores are $D1 = 0.02$, $D2=0.50$, $D3=0.32$ and $D4=0.17$. Final document ranking is shown in table 9.

Table 8. Limiting Super Matrix

	SE1	SE2	SE3	D1	D2	D3	D4
SE1	0.32	0.32	0.32	0.00	0.00	0.00	0.00
SE2	0.19	0.19	0.19	0.00	0.00	0.00	0.00
SE3	0.49	0.49	0.49	0.00	0.00	0.00	0.00
D1	0.00	0.00	0.00	0.02	0.02	0.02	0.02
D2	0.00	0.00	0.00	0.50	0.50	0.50	0.50
D3	0.00	0.00	0.00	0.32	0.32	0.32	0.32
D4	0.00	0.00	0.00	0.17	0.17	0.17	0.17

Table 9. Final Scores and Ranks

Document	D2	D3	D4	D1
Scores	0.5	0.32	0.17	0.02
Rank	1	2	3	4

3.4 Weighted Fuzzy ANP

The Fuzzy ANP model does not consider search engine importance weights in result merging. However, in metasearch, often the best performance is obtained when search engine importance weights are considered in result merging. Search engine importance weights can be assigned based on prior performance, reputation etc. The Weighted Fuzzy ANP model for metasearch was created as an extension to the Fuzzy ANP model, so as to consider search engine importance weights while merging result lists from them.

The Weighted Fuzzy ANP model is similar to the Fuzzy ANP model. The difference is that search engine importance weights are considered in pair-wise. Continuing from section 3.3, let us say w_1, w_2, \dots, w_n be the importance weights of the search engines, SE_1, SE_2, \dots, SE_n in SE-SET. Thus w_i be the importance weight assigned to the search engine SE_i . Let w_{max} and w_{min} be the highest and lowest importance weights assigned to the search engines. We can compare each pair of search engines based on Saty’s scale. So for a pair of search engines SE_i, SE_k with weights w_i, w_k respectively, we can compute $P(SE_i, SE_k)$ for all SE_i and SE_k such that $w_i > w_k$ and $w_i \neq w_k$ and $w_{max} \neq w_{min}$ based on equation (6). Automatically $P(SE_k, SE_i) = 1/P(SE_i, SE_k)$. When $w_i = w_k$ or $w_{max} = w_{min}$ then $P(SE_i, SE_k) = P(SE_k, SE_i) = 1$. For all pairs of search engines, (SE_i, SE_k) such that $SE_i, SE_k \in SE-SET$ we can calculate $P(SE_i, SE_k)$ and construct an $n \times n$ matrix. This can be normalized by dividing each element by its column sum and then averaging over rows to obtain a priority vector which contains pair-wise comparison weights for search engines. Let these weights be denoted by SW_1, SW_2, \dots, SW_n for search engines SE_1, SE_2, \dots, SE_n

respectively. In Step 2, in the Weighted Fuzzy ANP process, we compute $P(D_f, SE_i, SE_k)$ and $P(SE_f, D_i, D_k)$ as per equation (5) & (6). However, priority vector for any document $VD_f = [SW_1 \bullet SSE_{f1}, SW_2 \bullet SSE_{f2}, .. SW_n \bullet SSE_{fn}]$ and priority vector for search engine $SE_f, VSE_f = [SW_f \bullet SD_{f1}, SW_f \bullet SD_{f2}, .. SW_f \bullet SD_{fm}]$.

$$P(SE_i, SE_k) = \frac{w_i - w_k}{w_{max} - w_{min}} \bullet 9 \tag{6}$$

4 Experiments and Results

The focus of our experiments is to compare the performance of our proposed models, Fuzzy ANP and Weighted Fuzzy ANP versus the Borda-Fuse, Weighted Borda-Fuse and OWA models. We do this performance comparison for score-based result merging when document scores from search engines are available.

4.1 Experiments

We use the Hersh’s OHSUMED [6] collection constituted in LETOR 2 (Learning TO Rank) [8] dataset. LETOR provides (query, document) relevance measures and a host of other similarity scores based the features of the LETOR dataset. The OSHUMED collection provided as a part of LETOR 2 consists of 106 queries. The degree of for each query-document pair is pre-judged and categorized as 0 (non relevant), 1 (possibly relevant) and 2(definitely relevant). There are a total of 16,140 query-document pairs with relevance judgments. There are 25 features for each document and relevance scores between 0 and 1 with respect to a query. For our experiments features are treated as search systems and the result list of documents returned by them along with document scores for the 106 queries in the OHSUMED dataset are treated as result lists for merging.

Our metric for measuring performance is Recall-Based (RB) Precision. The detailed theory of RB precision can be found in [2]. RB precision is used in case when there are a series of documents ranked in a weak ordering and we need to find out the precision for various levels of recall. The formula is as described in equation 7.

$$\text{Precision} = \frac{x * n}{x * n + j + s * \frac{i}{r}} \tag{7}$$

Where (1) x is one of the standardized recall values i.e., 0.25, 0.5, 0.75, etc; (2) n is the number of relevant documents in the collection; (3) s is the number of relevant document needed to be retrieved from the final rank, where final rank refers to the rank containing the needed number of relevant documents for completing the total number of retrieved and relevant documents as specified by x *n; (4) i is the number of irrelevant documents in the final rank; (5) r is the number of relevant documents in the final rank; (6) j is the number of non-relevant documents retrieved from all the ranks above the final rank. We compute the precision at every level of recall and compute the average of these values to obtain the average precision. Average Precision is use as a performance measure.

Our objective is to gauge the performance of our proposed models Fuzzy ANP and Weighted Fuzzy ANP models with the performance of the Borda-Fuse and OWA models. For each iteration of experiment we randomly pick a number of search systems (N) to be merged. N varies between 2 and 12. We pick a query out of the 106 at random, and obtain result lists from the search engines selected in the previous step. We merge these result lists using the OWA/Borda-fuse/Weighted Borda Fuse/Fuzzy ANP/Weighted Fuzzy ANP models. The Weighted Borda-Fuse and Weighted Fuzzy ANP models for metasearch require search engine performance weights. We use odd queries in the collection of 106 queries to evaluate the performance weights in case of these models and even queries for experimentation.

For our models and the OWA model, we vary the Linguistic Quantifier parameter α , from 0.25 to 2, that is used to compute ordered weights in the OWA model and column stochastic weights in our proposed models. We calculate the RB-precision of the merged list from each of the models based on relevance judgments provided as part of the dataset for recall levels of 0.25, 0.5, 0.75 and 1 and compute the average. We perform 200 iterations of experiment for each value of N and α and average of over N and α .

4.2 Results

Table 10, shows the variation in average precision when the number of search engines (N) being merged is varied from 2 to 12. The benefits of metasearch are illustrated by the results as the overall average precision of the merged result list goes up when merging more number of search engines. Clearly the OWA model outperforms the Borda-Fuse and Weighted Borda-Fuse models for result merging. Also, our Fuzzy ANP model outperforms the Borda-Fuse model, the Weighted Borda-Fuse model and the OWA model by 57%,54% and 34% respectively. Our Weighted Fuzzy ANP model also outperforms the Borda-Fuse model, the Weighted Borda-Fuse model and the OWA model by 63%,60% and 39% respectively. The Weighted Fuzzy ANP model also improves on the performance of the Fuzzy ANP model by 5%.

Table 10. Comparing Model Performance vs. Number of Search Engine Result Lists Merged

Average Precision					
	Borda-Fuse	Weighted Borda-Fuse	OWA	Fuzzy ANP	Weighted Fuzzy ANP
2	0.3467	0.362	0.4051	0.5312	0.5508
4	0.3571	0.368	0.4237	0.5621	0.5793
6	0.3675	0.371	0.4297	0.5824	0.5987
8	0.3755	0.381	0.4332	0.5913	0.6193
10	0.3948	0.392	0.4681	0.6135	0.6378
12	0.4030	0.412	0.4732	0.6615	0.6889

Table 11 shows the variation in average precision when the Linguistic Quantifier parameter α used to compute weights is varied from 0.25 through to 2. Consistent with the findings of Diaz [5], the performance of the OWA model is best when $\alpha = 0.25$ and goes down to a lowest value when $\alpha = 1$. When α increases beyond that value the performance in terms of RB-precision goes up. The performance of the OWA model is poorest when ‘orness’ of aggregation is balanced i.e., under simple averaging conditions. Under conditions of high orness when $\alpha \leq 1$ and under high andness conditions when $\alpha \geq 1$, the model performance of the OWA model is higher. However, the performance of both the Fuzzy ANP and Weighted Fuzzy ANP models gradually go up when orness aggregation goes down i.e., as α progresses from 0.25 towards 2. The Fuzzy ANP model and Weighted Fuzzy ANP model both improve on the OWA model for all values of α .

Table 11. Comparing Model Performance over variation of Linguistic Quantifier Parameter α

Average Precision						
ALPHA	Borda-Fuse	Weighted Borda-Fuse	OWA	Fuzzy ANP	Weighted Fuzzy ANP	
0.25	0.3814	0.4011	0.4645	0.5412	0.5521	
0.5	0.3814	0.4011	0.4413	0.5767	0.5813	
1	0.3814	0.4011	0.3814	0.5971	0.6023	
2	0.3814	0.4011	0.4521	0.6243	0.6315	
2.5	0.3814	0.4011	0.4734	0.6473	0.6613	

5 Conclusions

In this paper we have proposed two models for result merging for metasearch that are based on the Analytical Network Process (ANP) that employs Fuzzy Linguistic Quantifiers to construct a column stochastic weighted super matrix for the convergence of the ANP process. We compare our model to three existing models for the result aggregation. The first of these is the non fuzzy result merging model called Borda Fuse. The second model is the Weighted Borda-Fuse model and the third model is the OWA model based on Yager’s [14] Ordered Weighted Average (OWA) operator. Our Fuzzy ANP model compares documents and search engines in pairs prior to merging using the ANP. It uses linguistic quantifier guided approach in transforming the ANP Super Matrix into the Weighted Super Matrix. One short coming of the Fuzzy ANP model is that it does not consider search engine prior performance in result merging. Previous work by Aslam [1] had shown the advantage of taking into consideration prior search engine performance. To overcome this we came up with an extension that was the Weighted Fuzzy ANP model for result merging. Our experiments show that our proposed models Fuzzy ANP and Weighted Fuzzy ANP both outperform the Borda-Fuse, Weighted Borda-Fuse and OWA models. Also the Weighted Fuzzy ANP model outperforms the Fuzzy ANP model for result merging.

References

1. Aslam, J., Montague, M.: Models for metasearch. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 275–284. ACM Press, New Orleans (2001)
2. Bollmann, P., Raghavan, V.V., Jung, G.S., Shu, L.C.: On probabilistic notions of precision as a function of recall. *Information Processing and Management* 28, 291–315 (1992)
3. Borda, J.C.: *Memoire sur les elections au scrutiny*. Histoire de l'Academie Royale des Sciences, Paris (1781)
4. Diaz, E.D., De, A., Raghavan, V.V.: A Comprehensive OWA-Based Framework for Result Merging in Metasearch. In: Ślęzak, D., Yao, J., Peters, J.F., Ziarko, W.P., Hu, X. (eds.) *RSFDGrC 2005*. LNCS (LNAI), vol. 3642, pp. 193–201. Springer, Heidelberg (2005)
5. Diaz, E.D.: *Selective Merging of Retrieval Results for Metasearch Environments*. University of Louisiana Press, Lafayette (2004)
6. Hersh, W., Buckley, C., Leone, T.J., Hickam, D.: OHSUMED: An interactive retrieval evaluation and new large test collection for research. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 275–284. ACM Press, Dublin (1994)
7. Hull, D.A., Pedersen, J.O., Schütze, H.: Method combination for document filtering. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 279–287. ACM Press, Zurich (1996)
8. Liu, T., Xu, J., Qin, T., Xiong, W., Li, H.: LETOR: Benchmark dataset for re-search on learning to rank for information retrieval. In: *LR4IR 2007 in Conjunction with SIGIR 2007*, pp. 1–6. ACM Press, Amsterdam (2007)
9. Meng, W., Yu, C., Liu, K.: Building efficient and effective metasearch engines. *ACM Computing Surveys* 34, 48–89 (2002)
10. Saaty, T.L.: *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation (Decision Making Series)*. McGraw-Hill, New York (1980)
11. Saaty, T.L.: Relative Measurement and its Generalization in Decision Making: Why Pair wise Comparisons are Central in Mathematics for the Measurement of Intangible Factors - The Analytic Hierarchy/Network Process. *Review of the Royal Spanish Academy of Sciences, Series A, Mathematics* 102, 251–318 (2007)
12. Saaty, T.L.: *Decision Making with Dependence and Feedback: The Analytic Network Process*. RWS Publications, Pittsburgh (1996)
13. Thompson, P.: A combination of expert opinion approach to probabilistic information retrieval, part 2: mathematical treatment of CEO model. *Information Processing and Management* 26, 383–394 (1990)
14. Yager, R.R.: On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics* 18, 183–190 (1988)
15. Yager, R.R.: Quantifier guided Aggregating using OWA operators. *International Journal of Intelligent Systems* 11, 183–190 (1998)
16. Zadeh, L.A.: A computational approach to fuzzy quantifiers in natural languages. *Computers and Mathematics with Applications* 9, 149–184 (1983)

On the Satisfiability and Validity Problems in the Propositional Gödel Logic*

Dušan Guller

Department of Applied Informatics, Comenius University,
Mlynská dolina, 842 48 Bratislava, Slovakia
guller@fmph.uniba.sk

Abstract. This paper addresses the satisfiability and validity problems of a formula in the propositional Gödel logic. Our approach is based on the translation of a formula to an equivalent *CNF* one which contains literals of the augmented form: either a or $a \rightarrow b$ or $(a \rightarrow b) \rightarrow b$, where a, b are propositional atoms or the propositional constants $0, 1$. Since the equivalent output *CNF* may be exponential in the size of an input formula, we improve the translation using interpolation rules so that output *CNF* formulae are in linear size with respect to input ones; however, not equivalent - only equisatisfiable. A *CNF* formula is further translated to an equisatisfiable finite order clausal theory which consists of order clauses - finite sets of order literals of the forms $a = b$ or $a < b$, where $=$ and $<$ are interpreted by the equality and strict linear order on $[0, 1]$, respectively. A variant of the *DPLL* procedure, operating on order clausal theories, is proposed. The *DPLL* procedure is proved to be refutation sound and complete for countable order clausal theories. Finally, the validity problem of a formula (tautology checking) is reduced to the unsatisfiability of a finite order clausal theory.

1 Introduction

Infinitely-valued logics have not yet been explored so widely as finitely-valued ones. It is not known any general approach as signed logic one in the finitely-valued case. The solution of the *SAT* and *VAL* problems strongly varies on a chosen infinitely-valued logic. The same holds for the translation of a formula to clause form, the existence of which is not guaranteed in general. The results in this area have been achieved in several ways, since infinite truth value sets form distinct algebraic structures. One approach may be based on the reduction from the infinitely-valued case to the finitely-valued one, as it has been done e.g. for the *VAL* problem in the propositional infinitely-valued Łukasiewicz logic in [15]. Another approach exploits the reduction of the *SAT* problem to mixed integer programming (*MIP*) [12, 14]. [3] investigates the *VAL* problem in the prenex fragment of the first-order Gödel logic enriched by the relativisation operator Δ , denoted as the prenex G_∞^Δ . At first, a variant of Herbrand's Theorem for the prenex G_∞^Δ is proved, which reduces the *VAL* problem of a formula in the prenex G_∞^Δ to the *VAL* problem of an open formula in G_∞^Δ . Then a chain normal form is defined using

* Partially supported by the grants VEGA 1/0688/10, VEGA 1/0726/09, and Slovak Literary Fund.

the formulae $\phi \prec \psi$, as an abbreviation for $\neg\Delta(\psi \rightarrow \phi)$, and $\phi \equiv_{\Delta} \psi$, as an abbreviation for $\Delta(\phi \rightarrow \psi) \wedge \Delta(\psi \rightarrow \phi)$. These formulae express the strict dense linear order with endpoints and equality on $[0, 1]$, which is not possible without Δ in G_{∞} . Further, a meta-level logic of order clauses is defined, which is a fragment of classical one. An order clause is a finite set of inequalities of the form either $A < B$ or $A \leq B$ where $<, \leq$ are meta-level predicate symbols and A, B are atoms of G_{∞}^{Δ} considered as meta-level terms. The semantics of the meta-level logic of order clauses is given by classical interpretations on $[0, 1]$, varying on assigned (truth) values to atoms of G_{∞}^{Δ} handled as meta-level terms, which are the strict dense linear order with endpoints on $[0, 1]$; $<$ is interpreted as the strict dense linear order with endpoints and \leq as its reflexive closure on $[0, 1]$. A formula in the prenex G_{∞}^{Δ} is valid if and only if a translation of it to the order clause form is unsatisfiable with respect to the semantics of the meta-level logic. The chaining calculi in [4][5] may be used for efficient deduction over order clauses.

In the paper, we investigate *SAT* and *VAL* problems of a formula in the propositional Gödel logic. Our approach is based on the translation of a formula to an equivalent *CNF* one, Lemma 1, Section 3, which contains literals of the augmented form: either a or $a \rightarrow b$ or $(a \rightarrow b) \rightarrow b$, where a, b are propositional atoms or the propositional constants $0, 1$. We consider a ground fragment of the first-order two-valued logic with equality and strict order. The syntax is given by a class of order clausal theories. An order clause is a finite set of order literals of the form: either $a = b$ or $a \prec b$. The semantics is given by a class of order interpretations. An order interpretation is a first-order two-valued interpretation such that its universum is $[0, 1]$, $=$ is interpreted as $=_{[0,1]}$, and \prec as $<_{[0,1]}$. For the purpose of solving the *SAT* problem, a *CNF* formula is translated to an equisatisfiable finite order clausal theory, Lemma 3, Section 3. The basis is the translation of a literal to an order clause: e.g. $a \rightarrow b$ is translated to $a \prec b \vee a = b$ or $(a \rightarrow b) \rightarrow b$ to $b \prec a \vee b = 1$. The trichotomy on order literals: either $a \prec b$ or $a = b$ or $b \prec a$, naturally invokes proposing a variant of the *DPLL* procedure with a trichotomy branching rule as an algorithm for deciding the satisfiability of a finite order clausal theory. The *DPLL* procedure is proved to be refutation sound, and complete in the finite case, Theorem 1, Section 4. By means of the compactness theorem, Theorem 2, Section 4, we obtain the refutational completeness in the countable case as well, Corollary 1, Section 4. In case of solving the *VAL* problem, we exploit the fact that a formula ϕ is a tautology (valid) if and only if the order formula $\phi \prec 1$ is unsatisfiable, Theorem 3, Section 5.

The paper is organised as follows. Section 2 gives the basic notions, notation, and useful properties concerning the propositional Gödel logic. Section 3 deals with clause form translation. In Section 4, we propose a variant of the *DPLL* procedure with a trichotomy branching rule and prove its refutational soundness, completeness, as well as the compactness theorem. Section 5 solves the *VAL* problem (tautology checking).

2 Propositional Gödel Logic

Throughout the paper, we shall use the common notions of propositional many-valued logics. The set of propositional atoms of Gödel logic will be denoted as *PropAtom*. By *PropForm* we designate the set of all propositional formulae of Gödel logic built up

from *PropAtom* using the propositional constants 0 , the false, 1 , the true, and the connectives \neg , negation, \wedge , conjunction, \vee , disjunction, \Rightarrow , implication. We shall assume that Gödel logic is interpreted by the standard \mathbf{G} -algebra

$$\mathbf{G} = ([0, 1], \leq, \vee, \wedge, \Rightarrow_{\mathbf{G}}, \bar{\ }^{\mathbf{G}}, 0, 1)$$

where \vee and \wedge denote the respective supremum and infimum operators on $[0, 1]$,

$$a \Rightarrow_{\mathbf{G}} b = \begin{cases} 1 & \text{if } a \leq b, \\ b & \text{else,} \end{cases}$$

$$\bar{a}^{\mathbf{G}} = \begin{cases} 1 & \text{if } a = 0, \\ 0 & \text{else.} \end{cases}$$

We recall that \mathbf{G} is a complete linearly ordered lattice algebra; the supremum operator \vee is commutative, associative, idempotent, monotone, 0 is its neutral element; the infimum operator \wedge is commutative, associative, idempotent, monotone, 1 is its neutral element;¹ the residuum operator $\Rightarrow_{\mathbf{G}}$ of \wedge satisfies the condition of residuation:

$$\text{for all } a, b, c \in \mathbf{G}, a \wedge b \leq c \iff a \leq b \Rightarrow_{\mathbf{G}} c; \quad (1)$$

the Gödel negation $\bar{\ }^{\mathbf{G}}$ satisfies the condition:

$$\text{for all } a \in \mathbf{G}, \bar{a}^{\mathbf{G}} = a \Rightarrow_{\mathbf{G}} 0; \quad (2)$$

and the following properties, which will be exploited later, hold:²

For all $a, b, c \in \mathbf{G}$,

$$a \vee b \wedge c = (a \vee b) \wedge (a \vee c), \quad (3) \text{ (distributivity of } \vee \text{ over } \wedge)$$

$$a \wedge (b \vee c) = a \wedge b \vee a \wedge c, \quad (4) \text{ (distributivity of } \wedge \text{ over } \vee)$$

$$a \Rightarrow_{\mathbf{G}} (b \vee c) = a \Rightarrow_{\mathbf{G}} b \vee a \Rightarrow_{\mathbf{G}} c, \quad (5)$$

$$a \Rightarrow_{\mathbf{G}} b \wedge c = (a \Rightarrow_{\mathbf{G}} b) \wedge (a \Rightarrow_{\mathbf{G}} c), \quad (6)$$

$$(a \vee b) \Rightarrow_{\mathbf{G}} c = (a \Rightarrow_{\mathbf{G}} c) \wedge (b \Rightarrow_{\mathbf{G}} c), \quad (7)$$

$$a \wedge b \Rightarrow_{\mathbf{G}} c = a \Rightarrow_{\mathbf{G}} c \vee b \Rightarrow_{\mathbf{G}} c, \quad (8)$$

$$a \Rightarrow_{\mathbf{G}} (b \Rightarrow_{\mathbf{G}} c) = a \wedge b \Rightarrow_{\mathbf{G}} c, \quad (9)$$

$$((a \Rightarrow_{\mathbf{G}} b) \Rightarrow_{\mathbf{G}} b) \Rightarrow_{\mathbf{G}} b = a \Rightarrow_{\mathbf{G}} b, \quad (10)$$

$$(a \Rightarrow_{\mathbf{G}} b) \Rightarrow_{\mathbf{G}} c = ((a \Rightarrow_{\mathbf{G}} b) \Rightarrow_{\mathbf{G}} b) \wedge (b \Rightarrow_{\mathbf{G}} c) \vee c, \quad (11)$$

$$(a \Rightarrow_{\mathbf{G}} b) \Rightarrow_{\mathbf{G}} 0 = ((a \Rightarrow_{\mathbf{G}} 0) \Rightarrow_{\mathbf{G}} 0) \wedge b \Rightarrow_{\mathbf{G}} 0. \quad (12)$$

¹ Using the commutativity, associativity, idempotence, monotonicity, neutral elements of \vee and \wedge will not be explicitly referred to.

² We assume the decreasing operator priority sequence $\bar{\ }^{\mathbf{G}}, \wedge, \Rightarrow_{\mathbf{G}}, \vee$, which enables writing order clauses without parentheses.

A valuation \mathcal{V} of propositional atoms is a mapping $\mathcal{V} : PropAtom \longrightarrow [0, 1]$. A partial valuation \mathcal{V} of propositional atoms with the domain $dom(\mathcal{V}) \subseteq PropAtom$ is a mapping $\mathcal{V} : dom(\mathcal{V}) \longrightarrow [0, 1]$. Let $atoms(\phi), atoms(T) \subseteq dom(\mathcal{V})$ in case of \mathcal{V} being a partial valuation. The truth value ϕ in \mathcal{V} , in symbols $\|\phi\|^\mathcal{V}$, is defined by the standard way; the propositional constants $0, 1$ are interpreted by $0, 1$, respectively, and the connectives by the respective operators on \mathbf{G} . \mathcal{V} is a (partial) propositional model of ϕ , in symbols $\mathcal{V} \models \phi$, iff $\|\phi\|^\mathcal{V} = 1$. \mathcal{V} is a (partial) propositional model of T , in symbols $\mathcal{V} \models T$, iff for all $\phi \in T$, $\mathcal{V} \models \phi$. ϕ is a propositional consequence of T , in symbols $T \models_P \phi$, iff for every propositional model \mathcal{V} of T , $\mathcal{V} \models \phi$. ϕ is equivalent to ϕ' , in symbols $\phi \equiv \phi'$, iff for every valuation \mathcal{V} , $\|\phi\|^\mathcal{V} = \|\phi'\|^\mathcal{V}$. $\phi \mid T$ is satisfiable iff there exists a propositional model of $\phi \mid T$. $\phi \mid T$ is equisatisfiable to $\phi' \mid T'$ iff $\phi \mid T$ is satisfiable if and only if $\phi' \mid T'$ is satisfiable.

Let X, Y, Z be sets, $Z \subseteq X$, and $f : X \longrightarrow Y$ a mapping. By $X \subseteq_{\mathcal{F}} Y$ we denote X is a finite subset of Y . We designate $\mathcal{P}(X) = \{x \mid x \subseteq X\}$, $\mathcal{P}(X)$ is the power set of X ; $\mathcal{P}_{\mathcal{F}}(X) = \{x \mid x \subseteq_{\mathcal{F}} X\}$, $\mathcal{P}_{\mathcal{F}}(X)$ is the set of all finite subsets of X ; $f[Z] = \{f(z) \mid z \in Z\}$, $f[Z]$ is called the image of Z with respect to f ; and $f|_Z = \{(z, f(z)) \mid z \in Z\}$, $f|_Z$ is the restriction of f onto Z . $f : \omega \longrightarrow Y$ is a sequence of Y iff f is a bijection.

3 Translation to Clausal Form

We propose translation of a formula to an equivalent *CNF* formula, Lemma [1](#). In contrast to two-valued logic, we have to consider an augmented set of literals appearing in *CNF* formulae. Let $l, \phi \in PropForm$. l is a literal iff either $l = a$ or $l = a \rightarrow b$ or $l = (a \rightarrow b) \rightarrow b$ where $a \in PropAtom$ and $b \in PropAtom \cup \{0\}$. ϕ is a conjunctive \mid disjunctive normal form, in symbols *CNF* \mid *DNF*, iff either $\phi = 0$ or $\phi = 1$ or $\phi = \bigwedge_{i \leq n} \bigvee_{j \leq m_i} l_j^i \mid \phi = \bigvee_{i \leq n} \bigwedge_{j \leq m_i} l_j^i$ where l_j^i are literals [3](#).

Lemma 1. *Let $\phi \in PropForm$. There exists a *CNF* $\psi \equiv \phi$.*

Proof. It is straightforward to prove that there exists $\vartheta \equiv \phi$ without any occurrence of \neg . The proof is by induction on the structure of ϕ using [\(2\)](#); every subformula of the form $\neg\varphi$ of ϕ is replaced with $\varphi \rightarrow 0 \equiv \neg\varphi$. We further prove the statement:

$$\text{There exists a } \textit{CNF} \psi \equiv \vartheta. \tag{13}$$

The proof is by induction on the structure of ϑ ; all the occurrences of \rightarrow in ϑ are pushed down and the resulting *CNF* ψ is recursively built up. The obvious cases are $\vartheta \in PropAtom \cup \{0, 1\}$ and $\vartheta = \vartheta_1 \wedge \vartheta_2$. In the case $\vartheta = \vartheta_1 \vee \vartheta_2$, the distributivity of \vee over \wedge , [\(3\)](#), is exploited.

Let $\vartheta = \vartheta_1 \rightarrow \vartheta_2$. Then, by induction hypothesis, there exist *CNF*'s $\psi_1 \equiv \vartheta_1$, $\psi_2 \equiv \vartheta_2$, and we distinguish three cases for ψ_1, ψ_2 . Case 1: either $\psi_1 = 0$ or $\psi_2 = 1$ is obvious; $\psi_1 \rightarrow \psi_2 \equiv 1$. Case 2: $\psi_1 = 1$ is also obvious; $\psi_1 \rightarrow \psi_2 \equiv \psi_2$. Case 3: neither $\psi_1 = 0$ nor $\psi_2 = 1$ nor $\psi_1 = 1$. Then $\psi_1 = \bigwedge_{i \leq n} \bigvee_{j \leq m_i} l_j^i$, l_j^i are literals,

³ Associativity of \wedge, \vee will not be explicitly referred to, and hence, $\bigwedge_{i \leq n} \phi_i, \bigvee_{i \leq n} \phi_i \in PropForm$ are written without parentheses.

and we get two cases for ψ_2 : either $\psi_2 = \bigwedge_{r \leq v} \bigvee_{s \leq u_r} k_s^r$, k_s^r are literals, or $\psi_2 = 0$. Using (6), (5), (8), (7), (3), in both the cases, there exists

$$\bigwedge_{\theta \leq \Theta} \bigvee_{\xi \leq \Xi_\theta} \lambda_\xi^\theta \rightarrow \kappa_\xi^\theta \equiv \psi_1 \rightarrow \psi_2 \stackrel{(IH)}{\equiv} \vartheta_1 \rightarrow \vartheta_2 = \vartheta, \quad (14)$$

λ_ξ^θ are literals, either κ_ξ^θ are literals or $\kappa_\xi^\theta = 0$. We show that

$$\text{for all } \theta \leq \Theta \text{ and } \xi \leq \Xi_\theta, \text{ there exists a DNF } \delta_\xi^\theta \equiv \lambda_\xi^\theta \rightarrow \kappa_\xi^\theta. \quad (15)$$

Let $\theta \leq \Theta$ and $\xi \leq \Xi_\theta$. We then distinguish nine cases for λ_ξ^θ and κ_ξ^θ . Case 3.1: $\lambda_\xi^\theta = a$ and $\kappa_\xi^\theta = b$, $a \in PropAtom$, $b \in PropAtom \cup \{0\}$. Hence, $\delta_\xi^\theta = a \rightarrow b = \lambda_\xi^\theta \rightarrow \kappa_\xi^\theta$ is a DNF. Case 3.2: $\lambda_\xi^\theta = a \rightarrow b$ and $\kappa_\xi^\theta = c$, $a \in PropAtom$, $b, c \in PropAtom \cup \{0\}$. Hence,

$$\delta_\xi^\theta = ((a \rightarrow b) \rightarrow b) \wedge (b \rightarrow c) \vee c \stackrel{(I1)}{\equiv} (a \rightarrow b) \rightarrow c = \lambda_\xi^\theta \rightarrow \kappa_\xi^\theta$$

is a DNF. Case 3.3: $\lambda_\xi^\theta = (a \rightarrow b) \rightarrow b$ and $\kappa_\xi^\theta = c$, $a \in PropAtom$, $b, c \in PropAtom \cup \{0\}$. Hence,

$$\begin{aligned} \delta_\xi^\theta &= (a \rightarrow b) \wedge (b \rightarrow c) \vee c \\ &\stackrel{(I0)}{\equiv} (((a \rightarrow b) \rightarrow b) \rightarrow b) \wedge (b \rightarrow c) \vee c \stackrel{(I1)}{\equiv} ((a \rightarrow b) \rightarrow b) \rightarrow c = \lambda_\xi^\theta \rightarrow \kappa_\xi^\theta \end{aligned}$$

is a DNF. Cases 3.4 – 3.9: either $\lambda_\xi^\theta = a$ or $\lambda_\xi^\theta = a \rightarrow b$ or $\lambda_\xi^\theta = (a \rightarrow b) \rightarrow b$, and $\kappa_\xi^\theta = \varphi \rightarrow d$ where either $\varphi = c$ or $\varphi = c \rightarrow d$, $a, c \in PropAtom$, $b, d \in PropAtom \cup \{0\}$. By Cases 3.1 – 3.3, there exists a DNF $\gamma_\xi^\theta \equiv \lambda_\xi^\theta \rightarrow d$, and

$$\delta_\xi^\theta = \gamma_\xi^\theta \vee \varphi \rightarrow d \equiv \lambda_\xi^\theta \rightarrow d \vee \varphi \rightarrow d \stackrel{(8)}{\equiv} \lambda_\xi^\theta \wedge \varphi \rightarrow d \stackrel{(9)}{\equiv} \lambda_\xi^\theta \rightarrow (\varphi \rightarrow d) = \lambda_\xi^\theta \rightarrow \kappa_\xi^\theta$$

is a DNF. So, the claim (15) holds. We get that there exists a CNF

$$\psi \stackrel{(3)}{\equiv} \bigwedge_{\theta \leq \Theta} \bigvee_{\xi \leq \Xi_\theta} \delta_\xi^\theta \stackrel{(15)}{\equiv} \bigwedge_{\theta \leq \Theta} \bigvee_{\xi \leq \Xi_\theta} \lambda_\xi^\theta \rightarrow \kappa_\xi^\theta \stackrel{(14)}{\equiv} \vartheta.$$

Thus, the claim (13) holds. The induction is completed. We conclude that there exists a CNF $\psi \stackrel{(13)}{\equiv} \vartheta \equiv \phi$. \square

In Lemma 1 we have laid no restrictions on the use of the distributivity law, (3), during translation to conjunctive normal form. Therefore the size of the output CNF may be exponential in the size of an input formula. To avoid this disadvantage, we propose translation to CNF via interpolation using new atoms, which produces CNF formulae in linear size. A similar approach exploiting the renaming subformulae technique can be found in [17,6,13,16,18]. By $p_j^i \in PropAtom$ we denote atoms not yet occurring

in the set of formulae in question. The empty sequence of symbols is denoted as ε . Let $\phi \in PropForm$. We define the size of ϕ by recursion on the structure of ϕ :

$$|\phi| = \begin{cases} 1 & \text{if } \phi \in PropAtom \cup \{0, 1\}, \\ |\phi_1| + 1 & \text{if } \phi = \neg\phi_1, \\ |\phi_1| + |\phi_2| + 1 & \text{if } \phi = \phi_1 \diamond \phi_2 \text{ where } \diamond \in \{\wedge, \vee, \rightarrow\}. \end{cases}$$

Let $\phi_j \in PropForm$ and $p_j^i \in PropAtom$. We denote

$$\begin{aligned} \varphi_j^i &= \begin{cases} \phi_j & \text{if } \phi_j \in PropAtom, \\ p_j^i & \text{if } \phi_j \notin PropAtom; \end{cases} \\ +\pi_j^i &= \begin{cases} \varepsilon & \text{if } \phi_j \in PropAtom, \\ p_j^i \rightarrow \phi_j & \text{if } \phi_j \notin PropAtom; \end{cases} \\ -\pi_j^i &= \begin{cases} \varepsilon & \text{if } \phi_j \in PropAtom, \\ \phi_j \rightarrow p_j^i & \text{if } \phi_j \notin PropAtom. \end{cases} \end{aligned}$$

Let $\phi_1, \phi_2 \in PropForm$ and $p_j^i \in PropAtom$. In Table 1, we introduce interpolation rules. Let $\phi \in PropForm$. ψ is a *CNF* of ϕ iff ψ is a *CNF* obtained from $p^i \wedge (p^i \rightarrow \phi)$ for some i by a finite derivation using the interpolation rules. We denote the set of all *CNF*'s of ϕ as $CNF(\phi)$. Let $f, g : M \rightarrow \mathbb{N}$. $f \in O(g)$ iff there exist $n_0, k \in \mathbb{N}$, and for every $m \in M$ such that $g(m) \geq n_0$, $f(m) \leq k \cdot g(m)$.

Lemma 2. *Let $\phi \in PropForm$. $CNF(\phi) \neq \emptyset$, and for all $\psi \in CNF(\phi)$, ψ is equisatisfiable to ϕ , $|\psi| \in O(|\phi|)$.*

Proof. The proof of $CNF(\phi) \neq \emptyset$ is by induction on the structure of ϕ . It is straightforward to prove that $p^i \wedge (p^i \rightarrow \phi)$ is equisatisfiable to ϕ ; for every interpolation rule, its antecedent is equisatisfiable to its consequent; if for every i , ψ_i is equisatisfiable to ϕ_i , then so is $\bigwedge_i \psi_i$ to $\bigwedge_i \phi_i$; there exists k such that for every interpolation rule, the size of its consequent is less than or equal to k times the size of its antecedent. Let $\psi \in CNF(\phi)$. Then there exist i, n , a finite derivation $\zeta_0 = p^i \wedge (p^i \rightarrow \phi), \dots, \zeta_n = \psi$, and k such that for all $j \leq n$, ζ_j is equisatisfiable to ϕ and $|\zeta_j| \leq k \cdot |\phi|$. The proof is by induction on n using the previous statements. \square

We further introduce a ground fragment of the first-order two-valued logic with equality and strict order. The syntax is given by a class of order clausal theories. We form order literals and clauses from $PropAtom \cup \{0, 1\}$, regarded as constants, using binary predicates $=$, equality, and \prec , strict order. l is an order literal iff either $l = a = b = b = a$; since equality is commutative by definition, we identify $a = b$ and $b = a$; or $l = a \prec b$ where $a, b \in PropAtom \cup \{0, 1\}$. An order clause is a finite set of order literals. An order clause $\{l_1, \dots, l_n\}$ is written in the form $l_1 \vee \dots \vee l_n$. The order clause \emptyset is called the empty clause and denoted as \square . An order clause $\{l\}$ is called a unit order clause and denoted as l if it does not cause the ambiguity with the denotation of the single literal l in a given context. We designate the set of order clauses as $OrdCl$. Let l, l_1, \dots, l_n be order

Table 1. Interpolation rules

Case:	Positive interpolation	
	Negative interpolation	
$\phi_1 \wedge \phi_2$	$\frac{p_0^i \rightarrow \phi_1 \wedge \phi_2}{(p_0^i \rightarrow \phi_1) \wedge (p_0^i \rightarrow \phi_2)}$	(16)
	$\frac{\phi_1 \wedge \phi_2 \rightarrow p_0^i}{(\varphi_1^i \rightarrow p_0^i \vee \varphi_2^i \rightarrow p_0^i) \wedge \neg \pi_1^i \wedge \neg \pi_2^i}$	(17)
$\phi_1 \vee \phi_2$	$\frac{p_0^i \rightarrow (\phi_1 \vee \phi_2)}{(p_0^i \rightarrow \varphi_1^i \vee p_0^i \rightarrow \varphi_2^i) \wedge \neg \pi_1^i \wedge \neg \pi_2^i}$	(18)
	$\frac{(\phi_1 \vee \phi_2) \rightarrow p_0^i}{\phi_1 \rightarrow p_0^i \wedge \phi_2 \rightarrow p_0^i}$	(19)
$\phi_1 \wedge \phi_2 \rightarrow 0$	$\frac{p_0^i \rightarrow (\phi_1 \wedge \phi_2 \rightarrow 0)}{(p_0^i \rightarrow 0 \vee \varphi_1^i \rightarrow 0 \vee \varphi_2^i \rightarrow 0) \wedge \neg \pi_1^i \wedge \neg \pi_2^i}$	(20)
	$\frac{(\phi_1 \wedge \phi_2 \rightarrow 0) \rightarrow p_0^i}{((\phi_1 \rightarrow 0) \rightarrow p_0^i) \wedge ((\phi_2 \rightarrow 0) \rightarrow p_0^i)}$	(21)
$(\phi_1 \vee \phi_2) \rightarrow 0$	$\frac{p_0^i \rightarrow ((\phi_1 \vee \phi_2) \rightarrow 0)}{(p_0^i \rightarrow (\phi_1 \rightarrow 0)) \wedge (p_0^i \rightarrow (\phi_2 \rightarrow 0))}$	(22)
	$\frac{((\phi_1 \vee \phi_2) \rightarrow 0) \rightarrow p_0^i}{((\varphi_1^i \rightarrow 0) \rightarrow 0 \vee (\varphi_2^i \rightarrow 0) \rightarrow 0 \vee p_0^i) \wedge \neg \pi_1^i \wedge \neg \pi_2^i}$	(23)
$(\phi_1 \wedge \phi_2 \rightarrow 0) \rightarrow 0$	$\frac{p_0^i \rightarrow ((\phi_1 \wedge \phi_2 \rightarrow 0) \rightarrow 0)}{(p_0^i \rightarrow ((\phi_1 \rightarrow 0) \rightarrow 0)) \wedge (p_0^i \rightarrow ((\phi_2 \rightarrow 0) \rightarrow 0))}$	(24)
	$\frac{((\phi_1 \wedge \phi_2 \rightarrow 0) \rightarrow 0) \rightarrow p_0^i}{(\varphi_1^i \rightarrow 0 \vee \varphi_2^i \rightarrow 0 \vee p_0^i) \wedge \neg \pi_1^i \wedge \neg \pi_2^i}$	(25)
$((\phi_1 \vee \phi_2) \rightarrow 0) \rightarrow 0$	$\frac{p_0^i \rightarrow (((\phi_1 \vee \phi_2) \rightarrow 0) \rightarrow 0)}{(p_0^i \rightarrow 0 \vee (\varphi_1^i \rightarrow 0) \rightarrow 0 \vee (\varphi_2^i \rightarrow 0) \rightarrow 0) \wedge \neg \pi_1^i \wedge \neg \pi_2^i}$	(26)
	$\frac{(((\phi_1 \vee \phi_2) \rightarrow 0) \rightarrow 0) \rightarrow p_0^i}{(((\phi_1 \rightarrow 0) \rightarrow 0) \rightarrow p_0^i) \wedge (((\phi_2 \rightarrow 0) \rightarrow 0) \rightarrow p_0^i)}$	(27)
$((\phi_1 \rightarrow 0) \rightarrow 0) \rightarrow 0$	$\frac{p_0^i \rightarrow (((\phi_1 \rightarrow 0) \rightarrow 0) \rightarrow 0)}{p_0^i \rightarrow (\phi_1 \rightarrow 0)}$	(28)
	$\frac{(((\phi_1 \rightarrow 0) \rightarrow 0) \rightarrow 0) \rightarrow p_0^i}{(\phi_1 \rightarrow 0) \rightarrow p_0^i}$	(29)
$((\phi_1 \rightarrow \phi_2) \rightarrow 0) \rightarrow 0, \phi_2 \neq 0$	$\frac{p_0^i \rightarrow (((\phi_1 \rightarrow \phi_2) \rightarrow 0) \rightarrow 0)}{(p_0^i \rightarrow 0 \vee \varphi_1^i \rightarrow 0 \vee (\varphi_2^i \rightarrow 0) \rightarrow 0) \wedge \neg \pi_1^i \wedge \neg \pi_2^i}$	(30)
	$\frac{(((\phi_1 \rightarrow \phi_2) \rightarrow 0) \rightarrow 0) \rightarrow p_0^i}{((\phi_1 \rightarrow 0) \rightarrow p_0^i) \wedge (((\phi_2 \rightarrow 0) \rightarrow 0) \rightarrow p_0^i)}$	(31)
$(\phi_1 \rightarrow \phi_2) \rightarrow 0, \phi_2 \neq 0$	$\frac{p_0^i \rightarrow ((\phi_1 \rightarrow \phi_2) \rightarrow 0)}{(p_0^i \rightarrow ((\phi_1 \rightarrow 0) \rightarrow 0)) \wedge (p_0^i \rightarrow (\phi_2 \rightarrow 0))}$	(32)
	$\frac{((\phi_1 \rightarrow \phi_2) \rightarrow 0) \rightarrow p_0^i}{(\varphi_1^i \rightarrow 0 \vee (\varphi_2^i \rightarrow 0) \rightarrow 0 \vee p_0^i) \wedge \neg \pi_1^i \wedge \neg \pi_2^i}$	(33)
$\phi_1 \rightarrow \phi_2, \phi_2 \neq 0$	$\frac{p_0^i \rightarrow (\phi_1 \rightarrow \phi_2)}{(p_0^i \rightarrow \varphi_2^i \vee \varphi_1^i \rightarrow \varphi_2^i) \wedge \neg \pi_1^i \wedge \neg \pi_2^i}$	(34)
	$\frac{(\phi_1 \rightarrow \phi_2) \rightarrow p_0^i}{((\varphi_1^i \rightarrow \varphi_2^i) \rightarrow \varphi_2^i \vee p_0^i) \wedge (\varphi_2^i \rightarrow p_0^i) \wedge \neg \pi_1^i \wedge \neg \pi_2^i}$	(35)

literals and $C, C' \in \text{OrdCl}$. By $l \vee C$ we denote $\{l\} \cup C$ where $l \notin C$. Analogously, by $\bigvee_{i=1}^n l_i \vee C$ we denote $\{l_1\} \cup \dots \cup \{l_n\} \cup C$ where for all $1 \leq i \neq i' \leq n$, $l_i \notin C$ and $l_i \neq l_{i'}$. By $C \vee C'$ we denote $C \cup C'$. C is a subclause of C' , in symbols $C \sqsubseteq C'$, iff $C \subseteq C'$. An order clausal theory is a set of order clauses. A unit order clausal theory is a set of unit order clauses. Let $T, T' \subseteq \text{OrdCl}$. By $\text{atoms}(C) \mid \text{atoms}(T) \subseteq \text{PropAtom}$ we denote the set of all the propositional atoms occurring in $C \mid T$.

The semantics is given by a class of order interpretations. An order interpretation \mathcal{I} with the domain $\text{dom}(\mathcal{I}) = \text{PropAtom}$ is a first-order two-valued interpretation such that $\mathcal{U}_{\mathcal{I}} = [0, 1]$, for all $a \in \text{PropAtom}$, $a^{\mathcal{I}} \in [0, 1]$, $0^{\mathcal{I}} = 0$, $1^{\mathcal{I}} = 1$, and $=^{\mathcal{I}} = =_{[0,1]}$, $<^{\mathcal{I}} = <_{[0,1]}$. A partial order interpretation \mathcal{I} with the domain $\text{dom}(\mathcal{I}) \subseteq \text{PropAtom}$ is an order interpretation such that for all $a \in \text{dom}(\mathcal{I})$, $a^{\mathcal{I}} \in [0, 1]$. An (partial) order interpretation \mathcal{I} is identified with the (partial) valuation $\mathcal{V}_{\mathcal{I}} : \text{dom}(\mathcal{V}_{\mathcal{I}}) \rightarrow [0, 1]$, $\mathcal{V}_{\mathcal{I}}(a) = a^{\mathcal{I}}$. Let $\text{atoms}(l)$, $\text{atoms}(C)$, $\text{atoms}(C')$, $\text{atoms}(T)$, $\text{atoms}(T') \subseteq \text{dom}(\mathcal{I})$. \mathcal{I} is a (partial) model of l , in symbols $\mathcal{I} \models l$, iff either for $l = a = b$, $a^{\mathcal{I}} =_{[0,1]} b^{\mathcal{I}}$, or for $l = a < b$, $a^{\mathcal{I}} <_{[0,1]} b^{\mathcal{I}}$. \mathcal{I} is a (partial) model of C , in symbols $\mathcal{I} \models C$, iff there exists $l \in C$ such that $\mathcal{I} \models l$. \mathcal{I} is a (partial) model of T , in symbols $\mathcal{I} \models T$, iff for all $C \in T$, $\mathcal{I} \models C$. Note that \square and T such that $\square \in T$ are unsatisfiable by definition. C' is an order consequence of C , in symbols $C \models_O C'$, iff for every model \mathcal{I} of C , $\mathcal{I} \models C'$. C is an order consequence of T , in symbols $T \models_O C$, iff for every model \mathcal{I} of T , $\mathcal{I} \models C$. T' is an order consequence of T , in symbols $T \models_O T'$, iff for every model \mathcal{I} of T , $\mathcal{I} \models T'$. $C \mid T$ is satisfiable iff there exists a model of $C \mid T$. $C' \mid T'$ is equisatisfiable to $C \mid T$ iff $C' \mid T'$ is satisfiable if and only if $C \mid T$ is satisfiable.

By *OrdPropForm* we designate the augmented set of all order propositional formulae built up from *PropAtom* using 0 , 1 , \neg , \wedge , \vee , \rightarrow , and $<$, $=$. Note that *OrdPropForm* \supseteq *PropForm* by definition, and all the notions and notation concerned with *PropForm* are straightforwardly extended to *OrdPropForm*.

Lemma 3. *Let ϕ be a conjunctive normal form. There exists $T_{\phi} \subseteq_{\mathcal{F}} \text{OrdCl}$ such that T_{ϕ} is equisatisfiable to ϕ .*

Proof. By the definition of *CNF*, we distinguish three cases for ϕ . Case 1: $\phi = 0$. Then ϕ is unsatisfiable and $T_{\phi} = \{\square\} \subseteq_{\mathcal{F}} \text{OrdCl}$ is unsatisfiable as well. So, the claim holds. Case 2: $\phi = 1$. Then ϕ is satisfiable and $T_{\phi} = \emptyset \subseteq_{\mathcal{F}} \text{OrdCl}$ is satisfiable as well. So, the claim holds. Case 3: $\phi = \bigwedge_{i \leq n} \bigvee_{j \leq m_i} l_j^i$, l_j^i are literals.

For all $i \leq n$ and $j \leq m_i$, there exists (36)

$$C_j^i \in \text{OrdCl} \text{ such that } C_j^i \text{ is equisatisfiable to } l_j^i.$$

The proof is by definition. We get five cases for l_j^i . Case 3.1: $l_j^i = a$, $a \in \text{PropAtom}$. Then $C_j^i = a = 1$. Case 3.2: $l_j^i = a \rightarrow 0$, $a \in \text{PropAtom}$. Then $C_j^i = a = 0$. Case 3.3: $l_j^i = a \rightarrow b$, $a \in \text{PropAtom}$, $b \in \text{PropAtom}$. Then $C_j^i = a < b \vee a = b$. Case 3.4: $l_j^i = (a \rightarrow 0) \rightarrow 0$, $a \in \text{PropAtom}$. Then $C_j^i = 0 < a$. Case 3.5: $l_j^i = (a \rightarrow b) \rightarrow b$, $a \in \text{PropAtom}$, $b \in \text{PropAtom}$. Then $C_j^i = b < a \vee b = 1$. So, the claim (36) holds. By (36), there exists $T_{\phi} \subseteq_{\mathcal{F}} \text{OrdCl}$ such that $T_{\phi} = \{\bigvee_{j \leq m_i} C_j^i \mid i \leq n\}$ is equisatisfiable to ϕ . □

4 DPLL Procedure

We devise a variant of the DPLL procedure over finite order clausal theories. Let l , l_1, l_2, l_3 be order literals. l is a contradiction iff either $l = 0 = 1$ or $l = 0 \prec 0$ or $l = 1 \prec 1$ or $l = a \prec 0$ or $l = 1 \prec a$ or $l = a \prec a$ where $a \in PropAtom$. l is a tautology iff either $l = 0 = 0$ or $l = 1 = 1$ or $l = 0 \prec 1$ or $l = a = a$ where $a \in PropAtom$. $l_1 \vee l_2 \vee l_3$ is a general trichotomy iff $l_1 = a \prec b$, $l_2 = a = b$, $l_3 = b \prec a$ where $a, b \in PropAtom \cup \{0, 1\}$. Let $T \subseteq OrdCl$. The basic rules are as follows:

(37) (One literal contradiction simplification rule)

$$\frac{T}{T \cup \{\square\}}$$

if T is a unit order clausal theory, $l \in T$, and l is a contradiction;

(38) (One literal transitivity rule of $=$ and \prec)

$$\frac{T}{T \cup \{a \diamond c\}} \quad \text{where } \diamond = \begin{cases} = & \text{if } \diamond_1 = \diamond_2 = =, \\ \prec & \text{else,} \end{cases}$$

if T is a unit order clausal theory, $a \diamond_1 b$, $b \diamond_2 c \in T$, and $\diamond_1, \diamond_2 \in \{=, \prec\}$;

(39) (General trichotomy branching rule)

$$\frac{T}{T - \{l_1 \vee C\} \cup \{l_1\} \mid T - \{l_1 \vee C\} \cup \{C\} \cup \{l_2\} \mid T - \{l_1 \vee C\} \cup \{C\} \cup \{l_3\}}$$

if $l_1 \vee C \in T$, $C \neq \square$, and $l_1 \vee l_2 \vee l_3$ is a general trichotomy.

Rule (39) reflects the linearity of $<_{[0,1]}$ in terms of general trichotomy. Rule (37) formalises its additional properties: the endpoints $0 <_{[0,1]} 1$ and strictness via contradictions. Rule (38) expresses the mutual transitivity of $=_{[0,1]}$ together with $<_{[0,1]}$. Rules (37), (38), (39) are sound in view of satisfiability:

T and $T \cup \{\square\}$ in the consequent of Rule (37) are both unsatisfiable. (40)

T is equisatisfiable to $T \cup \{a \diamond c\}$ in the consequent of Rule (38). (41)

Let \mathcal{I} be a partial order interpretation, $dom(\mathcal{I}) \supseteq atoms(T)$. (42)

$\mathcal{I} \models T$ if and only if $\mathcal{I} \models T - \{l_1 \vee C\} \cup \{l_1\}$ or $\mathcal{I} \models T - \{l_1 \vee C\} \cup \{C\} \cup \{l_2\}$

or $\mathcal{I} \models T - \{l_1 \vee C\} \cup \{C\} \cup \{l_3\}$ in the consequent of Rule (39).

(43)

T is satisfiable if and only if $T - \{l_1 \vee C\} \cup \{l_1\}$ or $T - \{l_1 \vee C\} \cup \{C\} \cup \{l_2\}$

or $T - \{l_1 \vee C\} \cup \{C\} \cup \{l_3\}$ in the consequent of Rule (39) is satisfiable.

The proof is by assumption and definition. The refutational completeness argument of the basic rules, Theorem [1](#)(ii), may be provided using the excess literal technique [2](#). From this point of view, Rules (37) and (38) handle the base case: T is a unit order clausal theory, while Rule (39) the induction one: it subtracts the excess literal measure of T at least by 1 for every clausal theory in a branch of its consequent.

T is closed under Rules (37) and (38) iff for every application of Rules (37) and (38) of the form $\frac{T}{T'}$, $T' = T$. By $\text{trans}(T) \subseteq \text{OrdCl}$ we denote the least set such that $\text{trans}(T) \supseteq T$ and $\text{trans}(T)$ is closed under Rules (37), (38).

Using the basic rules, one can construct a finitely generated tree with the input theory as the root in the usual manner, so as the classical *DPLL* procedure does; for every parent vertex, there exists an application of Rule (37) or (38) or (39) such that the parent vertex is the theory in its antecedent and the children vertices are the theories in its consequent. A branch of a tree is closed iff it contains a vertex T' such that $\square \in T'$. A branch of a tree is open iff it is not closed. A tree is closed iff every its branch is finite and closed. Note that a closed tree is finite by König's Lemma. A tree is open iff it is not closed. A tree is linear iff it consists of only one branch, beginning from its root and ending in its only leaf.

Lemma 4. *Let $T \subseteq \text{OrdCl}$.*

- (i) *If $T \subseteq_{\mathcal{F}} \text{OrdCl}$, then $\text{trans}(T) \subseteq_{\mathcal{F}} \text{OrdCl}$.*
- (ii) *If T is a unit order clausal theory and $\square \notin \text{trans}(T)$, then $\text{trans}(T)$ is a unit order clausal theory.*
- (iii) *$\text{atoms}(\text{trans}(T)) = \text{atoms}(T)$.*
- (iv) *$T \models_O \text{trans}(T)$.*
- (v) *If $T \subseteq_{\mathcal{F}} \text{OrdCl}$, then there exists a finite linear tree with the root T and the leaf $\text{trans}(T)$ constructed using Rules (37) and (38).*

Proof. By assumption and definition. □

The following lemma shows that Rules (37) and (38) are refutation complete for a countable unit order clausal theory, which will be exploited in the base case of Theorem [1](#)(ii) and in Theorem [2](#).

Lemma 5. *Let $T = \text{trans}(T) \subseteq \text{OrdCl}$ be a countable unit order clausal theory. There exists a partial model \mathfrak{A} of T , $\text{dom}(\mathfrak{A}) = \text{atoms}(T)$.*

Proof. By the theorem assumption that T is a unit order clausal theory, $\square \notin T = \text{trans}(T)$. In addition, by the theorem assumption that T is a countable set, there exists a sequence δ of $\text{atoms}(T)$. At first, we define partial order interpretations Mod_α by recursion on $\alpha \leq \omega$:

$$Mod_0 = \emptyset;$$

$$Mod_\alpha = Mod_{\alpha-1} \cup \{(\delta(\alpha-1), v_{\alpha-1})\} \quad (0 < \alpha < \omega),$$

$$\mathbb{M}_{\alpha-1} = \{\|a\|^{Mod_{\alpha-1}} \mid a \equiv \delta(\alpha-1) \in T, a \in dom(Mod_{\alpha-1}) \cup \{0, 1\}\},$$

$$\mathbb{S}_{\alpha-1} = \{Mod_{\alpha-1}(a) \mid a \prec \delta(\alpha-1) \in T, a \in dom(Mod_{\alpha-1})\},$$

$$\mathbb{I}_{\alpha-1} = \{Mod_{\alpha-1}(a) \mid \delta(\alpha-1) \prec a \in T, a \in dom(Mod_{\alpha-1})\},$$

$$v_{\alpha-1} = \begin{cases} \frac{\bigvee \mathbb{S}_{\alpha-1} \wedge \bigwedge \mathbb{I}_{\alpha-1}}{2}, & \mathbb{M}_{\alpha-1} = \emptyset, \\ \bigvee \mathbb{M}_{\alpha-1}, & \mathbb{M}_{\alpha-1} \neq \emptyset; \end{cases}$$

$$Mod_\omega = \bigcup_{\alpha < \omega} Mod_\alpha.$$

It is straightforward to prove the following statements:

$$\text{For all } \alpha \leq \omega, Mod_\alpha \text{ is a partial order interpretation, } dom(Mod_\alpha) = \delta[\alpha], \quad (44)$$

$$\text{and for all } \beta \leq \alpha, Mod_\beta \subseteq Mod_\alpha.$$

$$\text{For all } \alpha \leq \omega \text{ and } l \in T \text{ such that } atoms(l) \subseteq dom(Mod_\alpha), Mod_\alpha \models l. \quad (45)$$

$$\text{For all } \alpha \leq \omega \text{ and } a \in dom(Mod_\alpha), \text{ if } Mod_\alpha(a) = 0, \text{ then } a \equiv 0 \in T. \quad (46)$$

$$\text{For all } \alpha \leq \omega \text{ and } a \in dom(Mod_\alpha), \text{ if } Mod_\alpha(a) = 1, \text{ then } a \equiv 1 \in T. \quad (47)$$

The proofs are by induction on $\alpha \leq \omega$. We put $\mathfrak{A} = Mod_\omega$. By (44), $\mathfrak{A} = Mod_\omega$ is a partial order interpretation, $dom(\mathfrak{A}) = dom(Mod_\omega) \stackrel{(44)}{=} \delta[\omega] = atoms(T)$. Let $l \in T$. Then $atoms(l) \subseteq atoms(T) = dom(Mod_\omega) = dom(\mathfrak{A})$ and $\mathfrak{A} = Mod_\omega \stackrel{(45)}{\models} l$. So, $\mathfrak{A} \models T$. We conclude that \mathfrak{A} is a partial model of T , $dom(\mathfrak{A}) = atoms(T)$. \square

The DPLL procedure is refutation sound and complete.

Theorem 1 (Refutational Soundness and Completeness of the DPLL Procedure).

Let T be a countable order clausal theory.

- (i) If there exists a closed tree $Tree$ with the root T constructed using Rules (37), (38), (39), then T is unsatisfiable.
- (ii) If $T \subseteq_{\mathcal{F}} OrdCl$, then there exists a finite tree $Tree$ with the root T constructed using Rules (37), (38), (39) with the following properties:

$$\text{If } T \text{ is unsatisfiable, then } Tree \text{ is closed.} \quad (48)$$

$$\text{If } T \text{ is satisfiable, then } Tree \text{ is open and there exists} \quad (49)$$

$$\text{a partial model } \mathfrak{A} \text{ of } T, dom(\mathfrak{A}) = atoms(T), \text{ related to } Tree.$$

Proof. (i) The proof is by induction on the structure of $Tree$ using (40), (41), (42).

(ii) We distinguish two cases:

either $\square \in T$ or $\square \notin T$.

Case 1: $\square \in T$. Then T is unsatisfiable and $Tree = T$ is a closed tree with the root T . So, (48) holds and (49) holds trivially.

Case 2: $\square \notin T$. We exploit the excess literal technique to construct a finite tree $Tree$ with the root T using Rules (37), (38), (39). Let $T^F \subseteq_{\mathcal{F}} OrdCl$. We define $elmeasure(T^F) = (\sum_{C \in T^F} |C|) - |T^F|$. We proceed by induction on $elmeasure(T)$.

Let $elmeasure(T) = 0$. Then, by the assumption $\square \notin T$ and the definition of $elmeasure(T)$, T is a unit order clausal theory. By Lemma 4(v), there exists a finite linear tree $Tree$ with the root T and the leaf $trans(T)$ constructed using Rules (37) and (38). We get two cases:

either $\square \in trans(T)$ or $\square \notin trans(T)$.

Case 2.1: $\square \in trans(T)$. Then $Tree$ is a closed tree with the root T ; its only branch from T to $trans(T)$ is closed. Hence, by (i), T is unsatisfiable. So, (48) holds and (49) holds trivially.

Case 2.2: $\square \notin trans(T)$. Then $Tree$ is an open tree with the root T ; its only branch from T to $trans(T)$ is open. Since T is a unit order clausal theory, by Lemma 4(ii), we get $trans(T)$ is a unit order clausal theory, and by Lemma 5 for $trans(T)$, there exists a partial model \mathfrak{A} of $trans(T)$, $dom(\mathfrak{A}) = atoms(trans(T))$. Hence, \mathfrak{A} is a partial model of $T \subseteq trans(T)$, $dom(\mathfrak{A}) = atoms(trans(T)) \stackrel{\text{Lemma 4(iii)}}{=} atoms(T)$, related to $Tree$ and T is satisfiable. So, (49) holds and (48) holds trivially.

Let $elmeasure(T) = n > 0$ and the statement hold for all $T^F \subseteq_{\mathcal{F}} OrdCl$ such that $elmeasure(T^F) < n$. Since $elmeasure(T) > 0$, by the definition of $elmeasure(T)$, there exists $l_1 \vee C \in T$ such that $C \neq \square$. Let l_2, l_3 be order literals such that $l_1 \vee l_2 \vee l_3$ is a general trichotomy. This yields the application of Rule (39)

$$\frac{T}{(T - \{l_1 \vee C\}) \cup \{l_1\} \mid (T - \{l_1 \vee C\}) \cup \{C\} \cup \{l_2\} \mid (T - \{l_1 \vee C\}) \cup \{C\} \cup \{l_3\}}.$$

We denote $T_1 = (T - \{l_1 \vee C\}) \cup \{l_1\}$, $T_2 = (T - \{l_1 \vee C\}) \cup \{C\} \cup \{l_2\}$, $T_3 = (T - \{l_1 \vee C\}) \cup \{C\} \cup \{l_3\}$. Then $elmeasure(T_1) < elmeasure(T)$, $elmeasure(T_2) < elmeasure(T)$, $elmeasure(T_3) < elmeasure(T)$, and by induction hypothesis, there exist finite trees $Tree_1$ with the root T_1 , $Tree_2$ with the root T_2 , $Tree_3$ with the root T_3 constructed using Rules (37), (38), (39) such that (48) and (49) hold for $Tree_1$, $Tree_2$, $Tree_3$. This yields

$$Tree = \frac{T}{Tree_1 \mid Tree_2 \mid Tree_3}$$

is a finite tree with the root T constructed using Rules (37), (38), (39). We get two cases:

either T is unsatisfiable or T is satisfiable.

Case 3: T is unsatisfiable. Then, by (42), T_1, T_2, T_3 are unsatisfiable, and by (48) for $Tree_1, Tree_2, Tree_3$, $Tree_1, Tree_2, Tree_3$ are closed trees. Hence, $Tree$ is a closed tree. So, (48) holds and (49) holds trivially for $Tree$.

Case 4: T is satisfiable. Then, by (42), there exists $1 \leq i \leq 3$ such that T_i is satisfiable. Hence, by (49) for $Tree_i$, $Tree_i$ is an open tree and there exists a partial model \mathfrak{A}_i of T_i , $dom(\mathfrak{A}_i) = atoms(T_i)$, related to $Tree_i$. By the definition of T_i , $T_i \models_O T$. As $\{l_1, l_2, l_3\}$ is a trichotomy, $atoms(l_1) = atoms(l_2) = atoms(l_3)$ and $atoms(T_i) \subseteq atoms(T)$. We get $Tree$ is an open tree and $\mathfrak{A} = \mathfrak{A}_i \cup \{(p, 0) \mid p \in atoms(T) - atoms(T_i)\}$, $dom(\mathfrak{A}) = atoms(T)$, is a partial model of T related to $Tree$. So, (49) holds and (48) holds trivially for $Tree$. The induction is completed. \square

The refutational completeness of the DPLL procedure can be generalised to the case of a countable order clausal theory by means of the following compactness theorem. Let $T \subseteq OrdCl$ and $A \subseteq PropAtom$. We denote $T|_A = \{C \mid C \in T, atoms(C) \subseteq A\} \subseteq T$, $atoms(T|_A) \subseteq atoms(T) \cap A$.

Theorem 2 (Compactness Theorem). *Let $T \subseteq OrdCl$ be a countable order clausal theory and δ be a sequence of $atoms(T)$. If for every $\alpha < \omega$, there exists a partial model \mathfrak{A}_α of $T|_{\delta[\alpha]} \subseteq_{\mathcal{F}} T$, $dom(\mathfrak{A}_\alpha) = \delta[\alpha]$, then there exists a partial model \mathfrak{A} of T , $dom(\mathfrak{A}) = atoms(T)$.*

Proof. We are able to define unit order clausal theories $T_\alpha \subseteq_{\mathcal{F}} OrdCl$ by recursion on $\alpha < \omega$ with the following properties:

$$atoms(T_\alpha) \subseteq \delta[\alpha]; \tag{50}$$

$$T_\alpha = trans(T_\alpha); \tag{51}$$

$$\text{for all } \beta \leq \alpha, T_\beta \subseteq T_\alpha; \tag{52}$$

$$T_\alpha \models_O T|_{\delta[\alpha]}; \tag{53}$$

$$\text{for every } \alpha \leq \beta < \omega, \text{ there exists } \beta \leq \beta^* < \omega \text{ such that} \tag{54}$$

$$\mathfrak{A}_{\beta^*} \text{ is a partial model of } T|_{\delta[\beta]} - T|_{\delta[\alpha]} \cup T_\alpha;$$

$$\text{for all } \alpha < \omega, dom(\mathfrak{A}_{\beta^*}) = \delta[\beta^*] \supseteq \delta[\beta] \cup \delta[\alpha] \stackrel{(50)}{\supseteq} atoms(T|_{\delta[\beta]} - T|_{\delta[\alpha]} \cup T_\alpha). \tag{55}$$

We exploit the excess literal technique to define T_α using Rules (37), (38), (39) from $T|_{\delta[\alpha]} - T|_{\delta[\alpha-1]} \cup T_{\alpha-1}$.

We put $T_\omega = \bigcup_{\alpha < \omega} T_\alpha$. Since, for all $\alpha < \omega$, $T_\alpha \subseteq_{\mathcal{F}} OrdCl$ is a unit order clausal theory, we get $T_\omega \subseteq OrdCl$ is a countable unit order clausal theory,

$$atoms(T_\omega) = \bigcup_{\alpha < \omega} atoms(T_\alpha) \stackrel{(48)}{\subseteq} \bigcup_{\alpha < \omega} \delta[\alpha] \subseteq \delta[\omega] = atoms(T). \tag{56}$$

It is straightforward to prove that $T_\omega = trans(T_\omega)$. By Lemma 5 we close that there exists a partial model \mathfrak{A}_ω of T_ω , $dom(\mathfrak{A}_\omega) = atoms(T_\omega)$. We put

$$\begin{aligned} \mathfrak{A} &= \mathfrak{A}_\omega \cup \{(a, 0) \mid a \in atoms(T) - dom(\mathfrak{A}_\omega)\}, \\ dom(\mathfrak{A}) &= dom(\mathfrak{A}_\omega) \cup dom(\{(a, 0) \mid a \in atoms(T) - dom(\mathfrak{A}_\omega)\}) \\ &= atoms(T_\omega) \cup atoms(T) - atoms(T_\omega) \stackrel{(56)}{=} atoms(T). \end{aligned}$$

As \mathfrak{A}_ω is a partial order interpretation, by the definition of \mathfrak{A} , \mathfrak{A} is a partial order interpretation. We further show that $\mathfrak{A} \models T$. Let $C \in T$. Then $atoms(C) \subseteq_{\mathcal{F}} atoms(T) = \delta[\omega]$, there exists $\alpha < \omega$ such that $atoms(C) \subseteq \delta[\alpha]$, and $C \in T|_{\delta[\alpha]}$. We get $\mathfrak{A}_\omega \models T_\omega$,

by the definition of \mathfrak{A} , $\mathfrak{A} \supseteq \mathfrak{A}_\omega$, and $\mathfrak{A} \models T_\omega = \bigcup_{\alpha < \omega} T_\alpha \models_O T_\alpha \models_{\mathcal{O}} T|_{\delta[\alpha]} \ni C$. So, $\mathfrak{A} \models T$. We conclude that \mathfrak{A} is a partial model of T , $dom(\mathfrak{A}) = atoms(T)$. \square

Corollary 1 (Refutational Completeness of the DPLL Procedure (The Countable Case)). *Let T be a countable order clausal theory. If T is unsatisfiable, then there exists a closed tree $Tree$ with the root T constructed using Rules (37), (38), (39).*

Proof. An immediate consequence of Theorems 1 and 2. Let T be unsatisfiable and δ be a sequence of $atoms(T)$. Then, by Theorem 2, there exists $\alpha < \omega$ such that $T|_{\delta[\alpha]} \subseteq_{\mathcal{F}} OrdCl$ is unsatisfiable. Hence, by Theorem 1(ii) and (48), there exists a closed tree $Tree'$ with the root $T|_{\delta[\alpha]}$ constructed using Rules (37), (38), (39). We get that there exists a closed tree $Tree$ with the root $T \supseteq T|_{\delta[\alpha]}$ constructed using Rules (37), (38), (39). The proof is by induction on the structure of $Tree'$. \square

Concerning the SAT problem of a formula, we conclude.

Corollary 2. *Let $\phi \in PropForm$. There exist an equisatisfiable $T_\phi \subseteq_{\mathcal{F}} OrdCl$ to ϕ and a finite tree $Tree_\phi$ with the root T_ϕ constructed using Rules (37), (38), (39) with the following properties:*

$$\text{If } \phi \text{ is unsatisfiable, then } Tree_\phi \text{ is closed.} \quad (57)$$

$$\text{If } \phi \text{ is satisfiable, then } Tree_\phi \text{ is open and there exists} \quad (58)$$

$$\text{a partial model } \mathfrak{A}_\phi \text{ of } \phi, \text{ dom}(\mathfrak{A}_\phi) = atoms(\phi).$$

Proof. An immediate consequence of Lemma 3 and Theorem 1. \square

Note that the SAT problem of a finite theory can be reduced to the SAT one of a formula in the usual manner. Let $T = \{\phi_i \mid i \leq n\} \subseteq_{\mathcal{F}} PropForm$. Then $\phi = \bigwedge_{i \leq n} \phi_i \in PropForm$ is equisatisfiable to T .

5 Tautology Checking

One application of the DPLL procedure may be to tautology checking. Let $\phi \in PropForm$. ϕ is a tautology (valid) iff for every valuation \mathcal{V} , $\mathcal{V} \models \phi$. The VAL problem of a formula ϕ can be reduced to the unsatisfiability of the order formula $\phi \prec 1$ consequently translated to an equisatisfiable finite order clausal theory T_ϕ . Then the unsatisfiability of T_ϕ is decided by the DPLL procedure. This section provides the technical details of the reduction, Theorem 3. In addition to the properties stated in Section 2, the following ones hold:

For all $\phi_1, \phi_2 \in PropForm$ and $\psi_1, \psi_2, \psi_3 \in OrdPropForm$,

$$(\phi_1 \wedge \phi_2) \prec 1 \equiv \phi_1 \prec 1 \vee \phi_2 \prec 1, \quad (59)$$

$$(\phi_1 \vee \phi_2) \prec 1 \equiv \phi_1 \prec 1 \wedge \phi_2 \prec 1, \quad (60)$$

$$\psi_1 \vee \psi_2 \wedge \psi_3 = (\psi_1 \vee \psi_2) \wedge (\psi_1 \vee \psi_3). \quad (61)$$

Theorem 3 (Reduction Theorem). *Let $\phi \in PropForm$. There exists $T_\phi \subseteq_{\mathcal{F}} OrdCl$ such that T_ϕ is unsatisfiable if and only if ϕ is a tautology.*

Proof. By Lemma 1, there exists a conjunctive normal form ψ such that $\psi \equiv \phi$, and we distinguish tree cases:

$$\text{either } \psi = 0 \text{ or } \psi = 1 \text{ or } \psi = \bigwedge_{i \leq n} \bigvee_{j \leq m_i} l_j^i, l_j^i \text{ are literals.}$$

Case 1: $\phi \equiv \psi = 0$. Then ϕ is not a tautology and $T_\phi = \emptyset \subseteq_{\mathcal{F}} OrdCl$ is satisfiable. So, the claim holds.

Case 2: $\phi \equiv \psi = 1$. Then ϕ is a tautology and $T_\phi = \{\square\} \subseteq_{\mathcal{F}} OrdCl$ is unsatisfiable. So, the claim holds.

Case 3: $\phi \equiv \psi = \bigwedge_{i \leq n} \bigvee_{j \leq m_i} l_j^i, l_j^i$ are literals. Then

$$\phi \text{ is a tautology if and only if } \phi \prec 1 \in OrdPropForm \text{ is unsatisfiable;} \quad (62)$$

$$\phi \prec 1 \equiv \psi \prec 1 = \left(\bigwedge_{i \leq n} \bigvee_{j \leq m_i} l_j^i \right) \prec 1 \stackrel{(59)}{=} \stackrel{(60)}{=} \bigvee_{i \leq n} \bigwedge_{j \leq m_i} l_j^i \prec 1. \quad (63)$$

For all $i \leq n$ and $j \leq m_i$, there exists a conjunction of disjunctions

$$\text{of order literals } \delta_j^i \in OrdPropForm \text{ such that } \delta_j^i \text{ is equisatisfiable to } l_j^i \prec 1. \quad (64)$$

The proof is by definition. We get five cases for l_j^i : Case 3.1: $l_j^i = a, a \in PropAtom$. Then $\delta_j^i = a \prec 1$. Case 3.2: $l_j^i = a \rightarrow 0, a \in PropAtom$. Then $\delta_j^i = 0 \prec a$. Case 3.3: $l_j^i = a \rightarrow b, a \in PropAtom, b \in PropAtom$. Then $\delta_j^i = b \prec a$. Case 3.4: $l_j^i = (a \rightarrow 0) \rightarrow 0, a \in PropAtom$. Then $\delta_j^i = a = 0$. Case 3.5: $l_j^i = (a \rightarrow b) \rightarrow b, a \in PropAtom, b \in PropAtom$. Then $\delta_j^i = (a \prec b \vee a = b) \wedge b \prec 1$. So, the claim (64) holds. By (64) and (63),

$$\bigvee_{i \leq n} \bigwedge_{j \leq m_i} \delta_j^i \text{ is equisatisfiable to } \bigvee_{i \leq n} \bigwedge_{j \leq m_i} l_j^i \prec 1 \text{ and } \phi \prec 1. \quad (65)$$

Hence, there exists $\varphi \in OrdPropForm$ such that

$$\varphi = \bigwedge_{r \leq v} \bigvee_{s \leq u_r} \kappa_s^r \stackrel{(61)}{=} \bigvee_{i \leq n} \bigwedge_{j \leq m_i} \delta_j^i \quad (66)$$

where κ_s^i are order literals. By (66) and (65), there exists $T_\phi \subseteq_{\mathcal{F}} OrdCl$ such that

$$T_\phi = \left\{ \bigvee_{s \leq u_r} \kappa_s^r \mid r \leq v \right\} \text{ is equisatisfiable to } \varphi, \bigvee_{i \leq n} \bigwedge_{j \leq m_i} \delta_j^i, \text{ and } \phi \prec 1. \quad (67)$$

We close that T_ϕ is unsatisfiable $\stackrel{(67)}{\iff} \phi \prec 1$ is unsatisfiable $\stackrel{(62)}{\iff} \phi$ is a tautology. \square

6 Conclusions

We have investigated the satisfiability and validity problems of a formula in the propositional Gödel logic. The satisfiability problem has been solved via the translation of a formula to an equivalent *CNF* one, containing literals of the forms a , $a \rightarrow b$, or $(a \rightarrow b) \rightarrow b$. A *CNF* formula has further been translated to an equisatisfiable finite order clausal theory, which consists of order clauses with order literals of the forms $a = b$ or $a \prec b$. $=$ and \prec are interpreted by the equality and strict linear order on $[0, 1]$, respectively. The trichotomy on order literals: either $a \prec b$ or $a = b$ or $b \prec a$, has naturally led to a variant of the *DPLL* procedure with a trichotomy branching rule, which is refutation sound and complete in the case of countable order clausal theories; where the compactness theorem holds. We have reduced the validity problem of a formula to the unsatisfiability of a finite order clausal theory.

References

1. Aguzzoli, S., Ciabatonni, A.: Finiteness of infinite-valued Łukasiewicz logic. *Journal of Logic, Language and Information* 9, 5–29 (2000)
2. Anderson, R., Bledsoe, W.W.: A linear format for resolution with merging and a new technique for establishing completeness. *Journal of the ACM* 17(3), 525–534 (1970)
3. Baaz, M., Ciabatonni, A., Fermüller, C.: Herbrand's theorem for prenex gödel logic and its consequences for theorem proving. In: Nieuwenhuis, R., Voronkov, A. (eds.) *LPAR 2001. LNCS (LNAD)*, vol. 2250, pp. 201–215. Springer, Heidelberg (2001)
4. Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation* 4(3), 217–247 (1994)
5. Bachmair, L., Ganzinger, H.: Ordered chaining calculi for first-order theories of transitive relations. *Journal of the ACM* 45(6), 1007–1049 (1998)
6. Boy de la Tour, T.: An optimality result for clause form translation. *Journal of Symbolic Computation* 14(4), 283–301 (1992)
7. Davis, M., Putnam, H.: A computing procedure for quantification theory. *Communications of the ACM* 7, 201–215 (1960)
8. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. *Communications of the ACM* 5, 394–397 (1962)
9. Guller, D.: Binary resolution over complete residuated Stone lattices. *Fuzzy Sets and Systems* 159(9), 1031–1041 (2008)
10. Guller, D.: On the refutational completeness of signed binary resolution and hyperresolution. *Fuzzy Sets and Systems* 160(8), 1162–1176 (2009)
11. Guller, D.: A *DPLL* procedure for the propositional Gödel logic. In: *Proceedings of the ICFC Conference, INSTICC*, pp. 31–42 (2010)
12. Hähnle, R.: Many-valued logic and mixed integer programming. *Annals of Mathematics and Artificial Intelligence* 12(3,4), 231–264 (1994)
13. Hähnle, R.: Short conjunctive normal forms in finitely-valued logics. *Journal of Logic and Computation* 4(6), 905–927 (1994)
14. Hähnle, R.: Proof theory of many-valued logic - linear optimization - logic design: Connections and interactions. *Soft Computing - A Fusion of Foundations, Methodologies and Applications* 1(3), 107–119 (1997)

15. Mundici, D.: Satisfiability in many-valued sentential logic is NP-complete. *Theoretical Computer Science* 52, 145–153 (1987)
16. Nonnengart, A., Rock, G., Weidenbach, C.: On Generating Small Clause Normal Forms. In: Kirchner, C., Kirchner, H. (eds.) *CADE 1998*. LNCS (LNAI), vol. 1421, pp. 397–411. Springer, Heidelberg (1998)
17. Plaisted, D.A., Greenbaum, S.: A structure-preserving clause form translation. *Journal of Symbolic Computation* 2(3), 293–304 (1986)
18. Sheridan, D.: The optimality of a fast CNF conversion and its use with SAT. In: *Online Proceedings of International Conference on the Theory and Applications of Satisfiability Testing (2004)*, <http://www.satisfiability.org/SAT04/programme/114.pdf>

A Fuzzy Approach to Resource Aware Automatic Parallelization

T. Trigo de la Vega^{1,*}, P. Lopez-Garcia^{1,2}, and S. Muñoz-Hernández³

¹ The IMDEA Software Institute, Madrid, Spain

² Spanish Research Council (CSIC), Madrid, Spain

³ School of Computer Science, Technical University of Madrid, Madrid, Spain
{teresa.trigo,pedro.lopez}@imdea.org, susana@fi.upm.es

Abstract. Any realistic approach to automatic program parallelization must take into account practical issues related to the resource usage of parallel executions, such as the overheads associated with parallel tasks creation, migration of tasks to remote processors, and communication. The aim of granularity control techniques is avoiding such overheads undermining the benefits of parallel executions. For example, sufficient conditions have been proposed to ensure that the parallel execution of some given tasks will not take longer than their corresponding sequential execution. However, when the goal is to optimize the average execution time of several runs, such conditions can be very conservative, causing a loss in parallelization opportunities. To solve this problem, we have proposed novel conditions based on fuzzy logic and performed an experimental assessment with real programs. The results show that such conditions select the optimal type of execution in most cases and behave much better than the conservative conditions.

Keywords: Fuzzy logic application, Parallel computing, automatic Parallelization, Granularity control, Scheduling, complexity Analysis.

1 Introduction

There is an increasing need of automatic software development techniques and tools that can exploit the potential of current parallel and distributed architectures. Since writing parallel code is a complex, tedious and prone to errors task, several approaches to automatic parallelization have been proposed. Most of them have concentrated on proving the correctness of automatic parallelizations (i.e., that the same results as those of the corresponding sequential executions are obtained) and their theoretical efficiency (i.e., that no more work than the sequential executions is performed) [5]. However, little work has been done that takes into account practical issues related to the resource usage of parallel executions, such as the overheads associated with tasks creation, possible migration of tasks to remote processors, and the associated communication overheads. Clearly, any realistic approach has to take all of these practical overheads and the resource usage of parallel tasks into account in order to avoid that the benefits of parallel

* Current affiliation: Google Inc.

executions be undermined by such overheads. This is the objective of granularity control: decide whether to execute some given tasks in parallel or sequentially based on conditions related to the resource usage of such tasks and the mentioned overheads. Granularity control has been studied in the context of traditional [9][13], functional [7] and logic programming [2][16][12]. Since tasks resource usages and overheads are not in general computable before the execution of the tasks, we are forced to resort to approximated conditions for deciding the type of execution to be performed. Previous approaches [12] have proposed sufficient conditions to ensure that the parallel (respectively, sequential) execution will not take longer than the corresponding sequential (respectively, parallel) one. However, when the goal is to optimize the average execution time of several runs, such conditions can be very conservative, and if they are not met, a type of execution is performed by default (either sequential or parallel), even though the other type of execution is the optimal one. Thus, this causes a loss in optimization opportunities. For this reason, we have proposed a novel approach that applies fuzzy logic to evaluate “fuzzy” conditions that, although can entail eventual slowdowns in some executions, speedup the whole computation on average (always preserving correctness).

It is remarkable the originality of this approach that is betting for the expressiveness of fuzzy logic to improve the decision making in the field of automatic program optimization and, in particular, in automatic parallelization including granularity control.

Fuzzy Logic Programming. Fuzzy logic has been a very fertile area during the last years. Specially in the theoretical side, but also from the practical point of view. Fuzzy logic programming systems, which are specially interesting by their simplicity, replace their inference mechanism, SLD-resolution, with a fuzzy variant that is able to handle partial truth. Most of these systems implement the fuzzy resolution introduced by Lee in [10]: the *Prolog-Elf* system [8], the *FRIL Prolog* system [1] and the *F-Prolog* language [11].

One of the most promising fuzzy tools for Prolog was the *Fuzzy Prolog* system [3]. Fuzzy Prolog adds fuzziness to a Prolog compiler using $CLP(\mathcal{R})$ instead of implementing a new fuzzy resolution method, as other former fuzzy Prologs do. It represents intervals as constraints over real numbers and *aggregation operators* as operations with these constraints, so it uses the Prolog built-in inference mechanism to handle the concept of partial truth.

Fuzzy Prolog [3] offers many useful features, however its truth value representation, based on constraints, is too general, which makes it complex to be interpreted by regular users. For this reason, and in order to provide a tool for practical application, a simpler variant was developed, called *RFuzzy* [15]. In *RFuzzy*, the truth value is represented by a simple real number.

RFuzzy is implemented as a *Ciao Prolog* [6] package, since *Ciao Prolog* offers the possibility of dealing with a higher order compilation through the implementation of *Ciao* packages.

Thus, since *RFuzzy* provides many nice features that represent an advantage with respect to previous fuzzy tools to model real problems, we have chosen *RFuzzy* for the implementation of our prototype in this work.

2 The Granularity Control Problem

We start by discussing the basic issues to be addressed in our approach to granularity control. In particular, we discuss how conditions for deciding between parallel and sequential execution can be devised. We consider the generic execution model described in [12]. Let $g = g_1, \dots, g_n$ be a task such that subtasks g_1, \dots, g_n are candidates for parallel execution. T_s represents the cost (execution time) of the sequential execution of g and T_i represents the cost of the sequential execution of subtask g_i . There can be many different ways to execute g in parallel, involving different choices of scheduling, load balancing, etc., each having its own cost. To simplify the discussion, we will assume that T_p represents in some way all of the possible costs. More concretely, $T_p \leq T_s$ should be understood as “ T_s is greater or equal than any possible value for T_p .” We will also assume that the points of parallelization of g are fixed, and, without loss of generality, that no tests – such as, perhaps, “independence” tests [5] – other than those related to granularity control are necessary. Thus, the purpose of granularity control is to determine, based on some conditions related to the cost of tasks and parallel execution overheads, whether the g_i ’s should be executed in parallel or sequentially in order to optimize the execution time of their whole computation.

Performing an accurate granularity control at compile-time is difficult since most of the information needed, as for example, input data size, is only known at run-time. An useful strategy is to do as much work as possible at compile-time and postpone some final decisions to run-time. This can be achieved by generating at compile-time cost functions which estimate task costs as a function of input data sizes, which are then evaluated at run-time when such sizes are known. Then, after comparing costs of parallel and sequential executions, it can be determined which of these types of executions must be performed. An interesting goal is to ensure that $T_p \leq T_s$. In general, this condition cannot be determined before executing g , while granularity control should intuitively be carried out ahead of time. Thus, we are forced to use approximations. This is discussed in the two following sections.

3 The Conservative Approach

The approach proposed in [12] consists on using safe approximations, i.e., evaluating (simple) sufficient conditions to detect when either, the parallel or sequential execution is the optimal one. There are two strategies, each with a different type of execution performed by default when the sufficient conditions are not met. They are described in the following.

Parallelizing a Sequential Program. This strategy corresponds to the case where tasks are executed sequentially unless parallel execution can be shown not to be slower. The sufficient condition $T_p^u \leq T_s^l$ for proving $T_p \leq T_s$ is used, where T_p^u denotes an upper bound on T_p and T_s^l a lower bound on T_s . We will use the superscripts l and u to denote lower and upper bounds respectively throughout the paper. Thus, if such sufficient condition does not hold, the tasks are executed sequentially by default. We refer the reader to [14] and [12] for a full description of compile-time analysis that infer lower and upper bounds on sequential and parallel execution times as functions of input data sizes.

Sequentializing a Parallel Program. This is the converse strategy, where tasks are executed in parallel unless sequential execution can be shown not to be slower. In this case, if the sufficient condition $T_s^u \leq T_p^l$ for proving $T_s \leq T_p$ does not hold, the tasks are executed in parallel by default.

4 The Fuzzy Approach

In some scenarios, it is not allowed to perform parallelizations if they do not ensure any speedup. However, in most environments it is justified to sacrifice performance in some cases in order to improve the speedup on average. Thus our approach is to give up strictly ensuring that $T_p \leq T_s$ holds and to use some relaxed heuristics using fuzzy logic which selects the optimal type of execution in most cases, improving the average execution time of several runs.

We use as a decision criteria the formula $T_p \leq T_s$. It is easy to transform such formula into $1 \leq T_s/T_p$ or the equivalent $T_s/T_p \geq 1$. We are implicitly using a crisp criteria in the sense that we use an operator whose truth values are defined mathematically.

If we move to classical logic and want to represent the condition of parallelizing or not a set of subtasks using a logic predicate, we could define *greater/2* as a predicate of two arguments that is successful if the first one is greater than the second one and false otherwise. We could check the condition *greater*($T_s/T_p, 1$) or rename this condition to a logic predicate, *greater/1*, of arity 1 that compares its argument with 1, succeeds if it is greater than 1 and fails otherwise (i.e., *greater*1(1.8) succeeds, whereas *greater*1(0.8) fails). With the boolean condition represented by the predicate *greater/1* it is easy to follow the conservative approach presented in Section 3. We can see that the concept of being “greater than” is very strict in the sense that some cases in which the value is close to 1 are going to be considered false. Let us introduce the concept of truth value. Till now we have been using two truth values *true* and *false*, or 1 and 0. But if we introduce levels of truth we could for example provide for a logic predicate intermediate truth values in between 0 and 1. We have defined other predicates similar to *greater/1* that are more flexible in their semantics. They are *quite_greater/1* and *rather_greater/1*. They are described in Section 5.1 and illustrated in Fig. 1.

With this set of predicates we are going to define a fuzzy framework for the experimental possibilities of using a fuzzy criteria to take decisions about parallelization of tasks.

4.1 Decision Making

Instead of deciding about the goodness of the parallelizations depending on a crisp condition as in the conservative approach, in this paper we make the decision attending to a couple of certainty factors: *S*, the certainty factor that represents the preference (its truth value) for executing the sequential variant of a program, and *P*, the certainty factor that represents the preference for executing the parallel variant of such program. Both certainty factors are real numbers, $S, P \in [0, 1]$. We define different fuzzy heuristics for assigning a value to each certainty factor and we compare a set of them in Section 5.2 in order to choose the best model (which is done in Section 5.3).

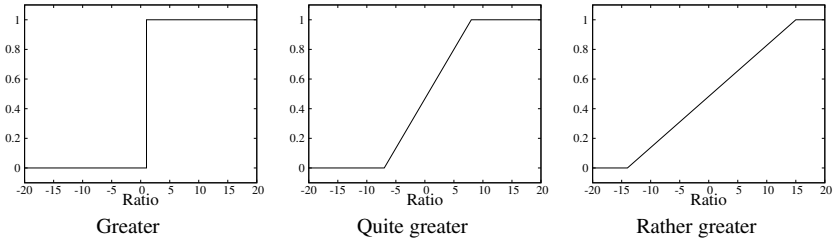


Fig. 1. Fuzzy sets for greater

Once the values of S and P have been already assigned, if $P > S$ then our task scheduling prototype executes the parallel variant of the program, otherwise it executes the sequential one.

5 Experimental Assessment

We have developed a prototype (Section 5.1) of a fuzzy task scheduler based on the approach described in Section 4. We have prepared a common framework to test the behavior of a set of different heuristics (Section 5.2) and we have also compared them with the rules of the conservative approach (Section 3) in order to be able to select the best results (Section 5.3). For a better understanding of these experiments, we present the behavior of our prototype for a progression of execution time data (Section 5.4). Finally, we have tested our prototype with real programs (Section 5.5) in order to demonstrate that it can be successfully applied in practice.

5.1 Prototype Implementation

All the granularity control methods presented in this paper have been implemented in *Ciao* [6], a multi-paradigm program development system that provides, among other features, a standard Prolog programming language. We have decided to use logic programming for implementing our approach because of its simplicity and for taking advantage of some useful extensions provided by the *Ciao Prolog* framework. In particular, the *Ciao* development system has integrated static analyzers for obtaining upper and lower bounds on execution times of procedures and tasks (which are part of the *CiaoPP* subsystem), and fuzzy packages for the calculation of certainty factors such as the *Rfuzzy* package which we have used for implementing the fuzzy logic rules.

As explained before, in our new approach to granularity control, the decision of how to execute a set of tasks is based on certainty factors associated to both, the sequential and parallel variants of their execution. So that, first of all, we have to quantify such certainty factors and then decide how to execute the tasks. The values of the certainty factors are provided by fuzzy rules that are able to combine fuzzy values using aggregation operators. According to *RFuzzy* syntax:

$$S(p, v_s) : op\ cond_1(v_1), cond_2(v_2), \dots, cond_n(v_n).$$

$$P(p, v_p) : op'\ cond'_1(v'_1), cond'_2(v'_2), \dots, cond'_n(v'_n).$$

The truth value v_s represents how much adequate is executing the program p sequentially, and is obtained by combining the truth values of the partial conditions v_1, \dots, v_n with the aggregation operator op . Similarly, v_p represents how much adequate is executing the program p in parallel. The bigger factor (S or P) will point out the selected execution (sequential or parallel).

In order to test the behavior of our method we have developed a set of conditions comparing a set of values of execution times $\{T_p^l, T_p^m, T_p^u, T_s^l, T_s^m, T_s^u\}$ by pairs (where the superscript m represents an average value). The comparison that each condition makes is calculated with the fuzzy relations *quite_greater* and *rather_greater*, whose definitions are (illustrated in Fig. 1):

$$quite_greater(X) = \begin{cases} 0 & \text{if } X \leq -7 \\ \frac{X+7}{15} & \text{if } -7 < X < 8 \\ 1 & \text{if } X \geq 8 \end{cases}$$

$$rather_greater(X) = \begin{cases} 0 & \text{if } X \leq -14 \\ \frac{X+14}{29} & \text{if } -14 < X < 15 \\ 1 & \text{if } X \geq 15 \end{cases}$$

We also use the *relative harmonic difference*, an experimental relation described in [14] as follows: $harmonic_diff(X, Y) = (X - Y) * (1/X + 1/Y)/2$. We have selected this relation because it compares two numbers in a relative and symmetric way, i.e.: $harmonic_diff(X, Y) = -harmonic_diff(Y, X)$. The *harmonic difference* only works well for positive numbers, but as we are working with execution times, it is enough for our purposes.

These fuzzy relations can be redefined with different bounds, although in this prototype we have only used the values 0, 7 and 14. These bounds have been selected according to the magnitude of the execution times that we provide for the programs (see Figures 2 to 7) in order to obtain significant results depending on the selected fuzzy relation.

5.2 Heuristic Comparison

We now discuss the assessment of our prototype using different aggregation operators. We have defined a benchmarks suite in terms of several characteristics. They are illustrated in Figures 2 to 7 which show the name of the program, lower bounds on sequential (T_s^l) and parallel (T_p^l) execution times, average sequential (T_s^m) and parallel (T_p^m) execution times, and upper bounds on sequential (T_s^u) and parallel (T_p^u) execution times. All the times are given in microseconds. Each figure illustrate both, parallel and sequential executions (in horizontal), and the intervals for all execution times defined by the mentioned upper and lower bounds (in vertical). The benchmarks have been created to cover all relevant cases in order to evaluate whether our fuzzy conditions select the optimal type of execution. Obviously, in real cases, the execution times

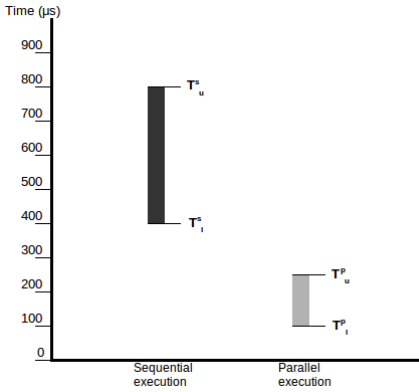


Fig. 2. Program p_1 . $T_s^l = 400$, $T_s^m = 600$, $T_s^u = 800$, $T_p^l = 100$, $T_p^m = 175$, $T_p^u = 250$.

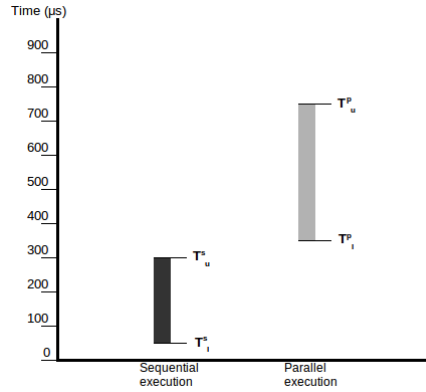


Fig. 3. Program p_2 . $T_s^l = 50$, $T_s^m = 175$, $T_s^u = 300$, $T_p^l = 350$, $T_p^m = 550$, $T_p^u = 750$.

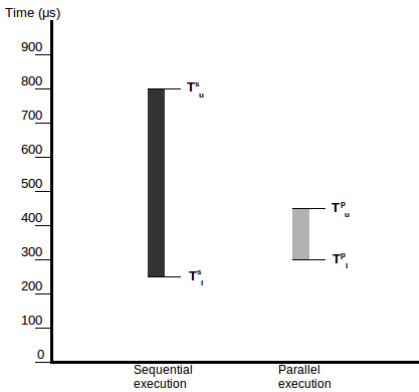


Fig. 4. Program p_3 . $T_s^l = 250$, $T_s^m = 525$, $T_s^u = 800$, $T_p^l = 300$, $T_p^m = 375$, $T_p^u = 450$.

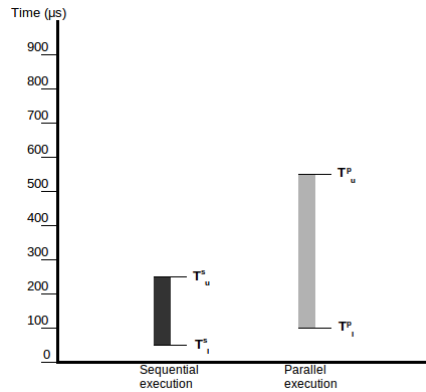


Fig. 5. Program p_4 . $T_s^l = 50$, $T_s^m = 150$, $T_s^u = 250$, $T_p^l = 100$, $T_p^m = 325$, $T_p^u = 550$.

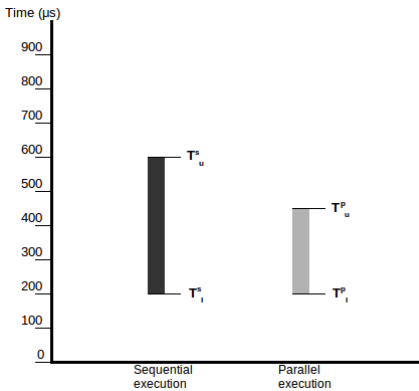


Fig. 6. Program p_5 . $T_s^l = 200$, $T_s^m = 400$, $T_s^u = 600$, $T_p^l = 200$, $T_p^m = 325$, $T_p^u = 450$.

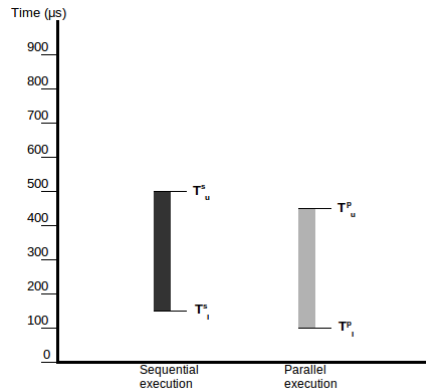


Fig. 7. Program p_6 . $T_s^l = 150$, $T_s^m = 325$, $T_s^u = 500$, $T_p^l = 100$, $T_p^m = 275$, $T_p^u = 450$.

Table 1. Aggregation operators execution times (in microseconds)

Prog.	Aggregation Operator		
	max	dprod	dluka
p_1	1.23	1.11	1.04
p_2	0.42	0.51	0.45
p_3	0.93	0.88	0.88
p_4	0.43	0.51	0.45
p_5	0.62	0.76	0.63
p_6	0.56	0.62	0.57
average	0.70	0.73	0.67

would be estimated at compile-time by a program analyzer like those of integrated in the *Ciao/CiaoPP* system [6,4,14]. For simplicity, we refer to the fuzzy set as gt , and to the *relative harmonic difference* relation as hd .

The rules of *fuzzy logic* for calculating the certainty factors P_i and S_i , for program p_i , $1 \leq i \leq 7$ (see Tables 2,3), have been composed using several aggregation operators but the results have shown that only the t-conorms max (max), Lukasiewicz (dluka) and sum (dprod) are correct (i.e., they always suggest the optimal type of execution) so we do not show the rest of the tested operations¹ in the results. We have seen how the three t-conorms max (max), Lukasiewicz (dluka) and sum (dprod) have the same behavior. Thus, in order to chose one of these aggregation operators, we have followed the criteria of the one more efficiently evaluated. In this sense, we have measured the execution time of evaluating the condition P_1 for each program using the three operators. These execution times have been obtained over an Intel platform (Intel Pentium 4 CPU 2.60GHz). They are shown in Table 1. The first column shows the name of the program (illustrated in Figures 2 to 7) and the three next ones, the aggregation operators. Each row shows the execution time (in microseconds) of the evaluation of the condition P_1 (see Tables 2,3) for the program using the three mentioned operators. The last row contains, for each operator, an average value on the execution time of evaluating such condition for all the programs. As we can see, the results are very similar for the aggregation operators max and dluka while for dprod they are almost always bigger. Although max is a little bit less efficient (on average) than dluka, max seems to be the best option due to its simplicity.

The whole set of proposed certainty factors and the results for each approach are shown in Tables 2,3. They correspond to the case of parallelizing a sequential program (i.e., where the action taken by default when there is no evidence towards executing in parallel is to execute sequentially). We have performed the experiments for two different levels of flexibility using the fuzzy relations *quite_greater* (Table 2) and *rather_greater* (Table 3). Both tables show similar data. The first column (Pr.) shows the name of the program. The second column (Op.) shows what would be the right (optimal) decision about the type of execution that should be performed, either parallel (Pa.) or sequential (Se.). The rest of the columns contain the results of evaluating the conditions. Columns 3 and 4 contain the results obtained using the conservative approach (Clas. Logic),

¹ The rest of the tested operations are: min, luka and prod.

while columns 5-18 contain the results obtained using our proposed conditions based on fuzzy logic (FUZZY LOGIC). Each column in the later group of columns corresponds to a different fuzzy condition. S_i and P_i , for $1 \leq i \leq 7$, are the truth values obtained for the certainty factors of the sequential and parallel executions of the program p_i respectively. The selected type of executions (using the process explained in Section 4.1, i.e., if $P_i > S_i$ then execute in parallel, otherwise execute sequentially) are highlighted. The decisions made by using the fuzzy conditions are always the optimal ones for these experiments. However, the conservative approach (*classical logic*) disagrees with the optimal ones in half of the cases.

For example, the condition $T_p^u \leq T_s^l$ holds for program p_1 (see Fig. 2), i.e., the truth value P_c takes the value 1. Thus, the parallel execution of p_1 is more efficient than the sequential one. In this case, both the *conservative approach (classical logic)* and the *fuzzy logic* approach agree in that the execution of p_1 should be in parallel. The converse condition ($T_s^u \leq T_p^l$) holds for program p_2 , i.e., the truth value S_c takes the value 1 (see Fig. 3), and thus, the optimal action is executing it sequentially. In this case, also both approaches agree in that the execution of p_2 should be sequentially.

For programs 3-6, both of the classical logic truth values (P_c and S_c) are always zero, which means that (by default) the suggested type of execution is *sequential* for all of these programs. However, from Figures 4, 5, 6 and 7 we can see that in most cases the optimal decision is to execute these programs in parallel. For example, consider program p_3 (see Fig. 4). We have that for this program, $T_p^u = 450 \mu s$ and $T_s^l = 250 \mu s$, and thus $T_p^u \leq T_s^l$ does not hold. The decision of executing p_3 *sequentially* made by *classical logic* is safe. However, in this case, since $T_s^u = 800 \mu s$, assuming that p_3 is run a significant number of times, we have that on average, executing p_3 in parallel would be more efficient than executing it sequentially. Unlike the classical logic conditions our proposed fuzzy approach selects the optimal type of execution for p_3 : its two subtasks should be executed in parallel. Program p_4 (see Fig. 5) represents the opposite case. In this case $T_s^u = 250 \mu s$ and $T_p^l = 100 \mu s$ so $T_s^u \leq T_p^l$ does not hold. But in this case $T_p^u = 550 \mu s$ and $T_s^l = 250 \mu s$. Thus, the best choice seems to be executing p_4 sequentially. Using classical logic, the selected execution is sequential, the one selected by default when none of the sufficient conditions P_c nor S_c hold. Using fuzzy logic the selected execution is also sequential. However our conditions provide enough evidences that support the decision of executing sequentially.

In the situations illustrated by programs p_5 and p_6 (see Figures 6 and 7) it is not so clear what type of execution should be selected. For program p_5 we have that $T_p^u = 450 \mu s$ and $T_s^l = 200 \mu s$. Thus, since the sufficient condition $T_p^u \leq T_s^l$ for executing in parallel does not hold, it seems that the program should be executed sequentially. However, since $T_p^l = 200 \mu s$ and $T_s^u = 600 \mu s$, the sufficient condition $T_s^u \leq T_p^l$ for executing sequentially does not hold either. Now, using our fuzzy logic approach, taking the four values T_p^l , T_p^u , T_s^l and T_s^u into account, a certainty factor of nearly 0.5 suggests that the best choice is to execute p_5 in parallel.

For program p_6 (see Fig. 7), none of the sufficient conditions $T_p^u \leq T_s^l$ and $T_s^u \leq T_p^l$ (for selecting parallel and sequential execution respectively) hold. However, since $T_p^u \leq T_s^u$ and $T_p^l \leq T_s^l$ hold, it is clear that the execution time of the sequential execution is going to belong to an interval whose limits are bigger than the limits of the parallel

Table 2. Selected types of executions using classical and fuzzy logic with *quite_greater*

Pr.	Op.	Clas. Logic		Fuzzy Logic													
		<i>greater</i>		<i>quite_greater</i>													
		Classical		Fuzzy 1		Fuzzy 2		Fuzzy 3		Fuzzy 4		Fuzzy 5		Fuzzy 6		Fuzzy 7	
P_c	S_c	P_1	S_1	P_2	S_2	P_3	S_3	P_4	S_4	P_5	S_5	P_6	S_6	P_7	S_7		
p_1	Pa.	1	0	0.73	0.48	0.73	0.48	0.73	0.48	0.57	0.35	0.57	0.35	0.57	0.35	0.57	0.36
p_2	Se.	0	1	0.48	0.93	0.49	0.93	0.49	0.93	0.34	0.58	0.33	0.59	0.34	0.58	0.31	0.58
p_3	Pa.	0	0	0.56	0.54	0.58	0.54	0.58	0.54	0.48	0.44	0.48	0.44	0.48	0.44	0.47	0.44
p_4	Se.	0	0	0.5	0.61	0.5	0.61	0.5	0.61	0.41	0.52	0.41	0.52	0.41	0.52	0.41	0.52
p_5	Pa.	0	0	0.54	0.53	0.55	0.53	0.55	0.53	0.47	0.45	0.47	0.45	0.47	0.45	0.47	0.45
p_6	Pa.	0	0	0.56	0.52	0.56	0.52	0.56	0.52	0.48	0.45	0.48	0.45	0.48	0.45	0.48	0.45

Table 3. Selected types of executions using classical and fuzzy logic with *rather_greater*

Pr.	Op.	Clas. Logic		Fuzzy Logic													
		<i>greater</i>		<i>rather_greater</i>													
		Classical		Fuzzy 1		Fuzzy 2		Fuzzy 3		Fuzzy 4		Fuzzy 5		Fuzzy 6		Fuzzy 7	
P_c	S_c	P_1	S_1	P_2	S_2	P_3	S_3	P_4	S_4	P_5	S_5	P_6	S_6	P_7	S_7		
p_1	Pa.	1	0	0.62	0.49	0.62	0.49	0.62	0.49	0.53	0.42	0.53	0.42	0.53	0.42	0.54	0.42
p_2	Se.	0	1	0.49	0.72	0.49	0.72	0.49	0.72	0.41	0.54	0.41	0.55	0.41	0.54	0.4	0.54
p_3	Pa.	0	0	0.53	0.52	0.54	0.52	0.54	0.52	0.49	0.47	0.49	0.47	0.49	0.47	0.48	0.47
p_4	Se.	0	0	0.5	0.55	0.5	0.55	0.5	0.55	0.45	0.51	0.45	0.51	0.45	0.51	0.45	0.51
p_5	Pa.	0	0	0.52	0.51	0.52	0.51	0.52	0.51	0.48	0.47	0.48	0.47	0.48	0.47	0.48	0.47
p_6	Pa.	0	0	0.53	0.51	0.53	0.51	0.53	0.51	0.49	0.47	0.49	0.47	0.49	0.47	0.49	0.47

Conditions (certainty factors):

$$\begin{aligned}
 P_c &= T_p^u \leq T_s^l \\
 S_c &= T_s^u \leq T_p^l \\
 P_1 &= \max(gt(T_s^l/T_p^u), gt(T_s^l/T_p^l), gt(T_s^m/T_p^m)) \\
 S_1 &= \max(gt(T_p^l/T_s^u), gt(T_p^l/T_s^l), gt(T_p^m/T_s^m)) \\
 P_2 &= \max(gt(T_s^l/T_p^u), gt(T_s^l/T_p^l), gt(T_s^u/T_p^u)) \\
 S_2 &= \max(gt(T_p^l/T_s^u), gt(T_p^l/T_s^l), gt(T_p^u/T_s^u)) \\
 P_3 &= \max(gt(T_s^l/T_p^u), gt(T_s^l/T_p^l), gt(T_s^m/T_p^m), gt(T_s^u/T_p^u)) \\
 S_3 &= \max(gt(T_p^l/T_s^u), gt(T_p^l/T_s^l), gt(T_p^m/T_s^m), gt(T_p^u/T_s^u)) \\
 P_4 &= \text{rel_hd}(0.5 * \text{hd}(T_s^m, T_p^m) + 0.25 * \text{hd}(T_s^u, T_p^u) + 0.25 * \text{hd}(T_s^l, T_p^l)) \\
 S_4 &= \text{rel_hd}(0.5 * \text{hd}(T_p^m, T_s^m) + 0.25 * \text{hd}(T_p^u, T_s^u) + 0.25 * \text{hd}(T_p^l, T_s^l)) \\
 P_5 &= \text{rel_hd}((\text{hd}(T_s^m, T_p^m) + \text{hd}(T_s^u, T_p^u) + \text{hd}(T_s^l, T_p^l))/3) \\
 S_5 &= \text{rel_hd}((\text{hd}(T_p^m, T_s^m) + \text{hd}(T_p^u, T_s^u) + \text{hd}(T_p^l, T_s^l))/3) \\
 P_6 &= \text{rel_hd}(0.25 * \text{hd}(T_s^m, T_p^m) + 0.5 * \text{hd}(T_s^u, T_p^u) + 0.25 * \text{hd}(T_s^l, T_p^l)) \\
 S_6 &= \text{rel_hd}(0.25 * \text{hd}(T_p^m, T_s^m) + 0.5 * \text{hd}(T_p^u, T_s^u) + 0.25 * \text{hd}(T_p^l, T_s^l)) \\
 P_7 &= \text{rel_hd}(0.25 * \text{hd}(T_s^m, T_p^m) + 0.25 * \text{hd}(T_s^u, T_p^u) + 0.5 * \text{hd}(T_s^l, T_p^l)) \\
 S_7 &= \text{rel_hd}(0.25 * \text{hd}(T_p^m, T_s^m) + 0.25 * \text{hd}(T_p^u, T_s^u) + 0.5 * \text{hd}(T_p^l, T_s^l))
 \end{aligned}$$

execution. Thus, is it more likely that the execution time of the parallel execution be less than the execution time of the sequential one, so that the right decision seems to execute p_6 in parallel. We can see that our proposed fuzzy conditions also suggests the parallel execution.

Finally, we can see that in those cases in which classical logic suggests a type of execution (sequential or parallel) with truth value 1, our fuzzy logic approach suggests the same type of execution too.

5.3 Selected Fuzzy Condition

Tables 2-3 show that all the fuzzy conditions (Fuzzy 1-7) select the same type of execution (sequential or parallel) independently of the fuzzy set used, either *quite_greater* or *rather_greater*. Our goal is to use the best (according to some criteria) fuzzy condition able to detect those situations where the parallel execution is faster than the sequential one on average, such that a conservative (safe) approach is not able to detect it but the fuzzy approach is. Approaches Fuzzy 1, 2 and 3 suggest parallel execution with greater evidence than Fuzzy 4, 5, 6 and 7 for both fuzzy sets, *quite_greater* (Table 2) and *rather_greater* (Table 3). As we are interested in suggesting to execute in parallel with evidences as bigger as possible we rule out the later subset of conditions and we focus our attention in the former set. Now, both Fuzzy 2 and 3 obtain the same values in all cases. Furthermore they provide higher evidences for parallel execution than the condition Fuzzy 1. This fact can be seen in programs p_3 and p_5 . As Fuzzy 2 is a subset of Fuzzy 3, evaluating the first one is more efficient than the second one (the Fuzzy 3 condition has one more comparison). Thus, the condition that we have selected is Fuzzy 2:

$$P_2 = \max(gt(T_s^l/T_p^u), gt(T_s^l/T_p^l), gt(T_s^u/T_p^u))$$

This condition obtains a better average case behavior by relaxing decision conditions (and possibly losing some precision in some cases). There may be cases in which our approach will select the slowest execution, however it will select the fastest one in a bigger number of cases. This tradeoff between safety and efficiency makes this new approach only applicable to non-critical systems, where no constraints about execution times must be met, and a wrong decision in a particular case will only cause a slow-down which is admissible. The fuzzy approach for sequentializing a parallel program is symmetric to the problem of parallelizing a sequential program. The condition that we have selected for the former is:

$$S_2 = \max(gt(T_p^l/T_s^u), gt(T_p^l/T_s^l), gt(T_p^u/T_s^u))$$

5.4 Decisions Progression

Focusing on program p_3 and using the fuzzy set *quite_greater* with the selected fuzzy model (in Section 5.3) we have developed an incremental experiment whose results are shown in Table 4. The main goal is to see how with this fuzzy logic approach we can select the optimal execution in those cases in which the conservative approach is not able to give a conclusion, and also, how our fuzzy logic approach detects all situations (safely) detected optimal by the conservative approach. Fig. 8 shows all the execution scenarios. The sequential execution times are fixed, while the parallel execution ones depend on each scenario. The later are represented by pairs $(T_p^l(i), T_p^u(i))$ where i is the

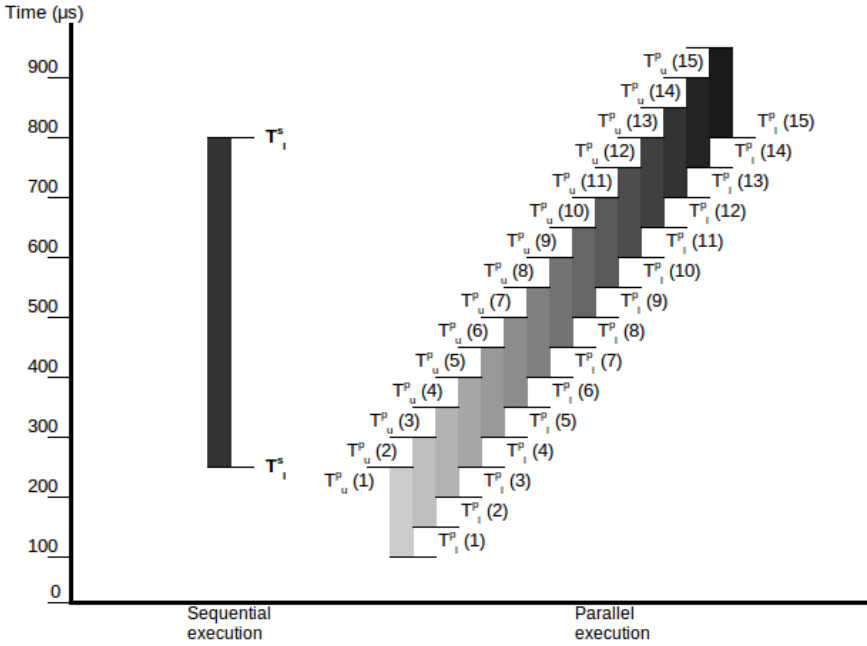


Fig. 8. Progression of executions of the example program p_3

Table 4. Progression of decisions using the fuzzy set *quite_greater*

Execution	Op.	Clas. Logic <i>greater</i>		Fuzzy Logic <i>quite_greater</i>	
		Classical		Fuzzy 2	
		P_c	S_c	P_2	S_2
p3_execution1	Pa.	1	0	0.68	0.49
p3_execution2	Pa.	0	0	0.64	0.5
p3_execution3	Pa.	0	0	0.61	0.52
p3_execution4	Pa.	0	0	0.6	0.53
p3_execution5	Pa.	0	0	0.58	0.54
p3_execution6	Pa.	0	0	0.57	0.56
p3_execution7	Se.	0	0	0.56	0.57
p3_execution8	Se.	0	0	0.55	0.58
p3_execution9	Se.	0	0	0.54	0.6
p3_execution10	Se.	0	0	0.54	0.61
p3_execution11	Se.	0	0	0.53	0.62
p3_execution12	Se.	0	0	0.53	0.64
p3_execution13	Se.	0	0	0.52	0.65
p3_execution14	Se.	0	0	0.52	0.66
p3_execution15	Se.	0	1	0.52	0.68

concrete case. The parallel execution times of each scenario are the times of the previous one plus 50 units, in order to appreciate the progression. The times of the first scenario are $T_p^l(1) = 100 \mu s$ and $T_p^u(1) = 250 \mu s$. P_c , S_c , P_2 and S_2 are defined as in Tables 2-3. According to classical logic we can see how only when $P_c = 1$ or $S_c = 1$ we obtain a justified answer (i.e., that the program must be executed in parallel or sequentially respectively). In the rest of the cases the selected type of execution is *sequential* by default, since we are following the philosophy of parallelizing a sequential program, and there are no evidences towards either type of execution. On the other hand, fuzzy logic always selects the optimal execution supported by evidences.

5.5 Experiments with Real Programs

The former experiments (Section 5.2) have shown that our fuzzy granularity control framework is able to capture which is the optimal type of execution on average. Moreover, in order to ensure that our approach can be applied in practice, we have performed some experiments with real programs (and real execution times). The experimental assessment have been made over an UltraSparc-T1, 8 cores x 1GHz (4 threads per core), 8GB of RAM, SunOS 5.10.

We have tested the *fuzzy model* selected in Section 5.3, so that only upper and lower bounds on (parallel and sequential) execution times were needed. Sequential execution times have been measured directly over the execution platform (executing the worst and best possible cases) while the parallel ones have been estimated as we explain in the following.

There are different ways of executing a task in parallel depending on the scheduling. The highest parallel execution time will be the one with the worst scheduling (i.e., the one in which the cores are idle as much as possible). Consider a task $g = g_1, \dots, g_n$ such that subtasks g_1, \dots, g_n are candidates for parallel execution. Assume that the number of cores of the processor is denoted as p , and the relation $\lceil n/p \rceil$ is denoted as k . We consider two different overheads of parallel execution: (a) the time needed for creating n parallel tasks, called $Create(n)$, and (b) an upper bound on the time taken from the point in which a parallel subtask g_i is created until its execution is started by a processor, denoted as $SysOverhead_i$. Both types of overheads have been experimentally measured for the execution platform. For the first one, we have measured directly the time of creating p threads. The second one has been obtained by using the expression $(Seq/2) - Par$, where Seq and Par are the measured execution times of a program consisting of two perfectly balanced tasks running with one and two threads respectively. Assume that T_{s_i} represents the cost (execution time) of the execution of subtask g_i and that $T_{s_1}, T_{s_2}, \dots, T_{s_n}$ are in descending order of cost. Then, we can estimate lower and upper bounds on the parallel execution time of task g (T_p^l and T_p^u respectively) as follows:

$$T_p^l = T_s^l/p \tag{1}$$

$$T_p^u = Create(p) + \sum_{i=1}^k (SysOverhead_i + T_{s_i}^u) \tag{2}$$

Table 5. Selected executions for real programs using the fuzzy set *quite_greater*

Execution	Op.	Clas. Logic		Fuzzy Logic		Speedup
		<i>greater</i>		<i>quite_greater</i>		
		Classical		Fuzzy 2		
		P_c	S_c	P_2	S_2	
<i>qsort(250)</i>	Pa.	0	0	0.6	0.53	1.66
<i>qsort(500)</i>	Pa.	0	0	0.6	0.53	1.74
<i>qsort(750)</i>	Pa.	0	0	0.6	0.53	1.74
<i>qsort(1000)</i>	Pa.	0	0	0.6	0.53	1.75
<i>qsort(1250)</i>	Pa.	0	0	0.6	0.53	1.71
<i>substitute(0)</i>	Se.	0	0	0.53	0.53	1.0
<i>substitute(10)</i>	Se.	0	0	0.6	0.65	1.0
<i>substitute(20)</i>	Se.	0	0	0.6	0.59	0.97
<i>substitute(30)</i>	Pa.	0	0	0.6	0.57	1.09
<i>substitute(40)</i>	Pa.	0	0	0.6	0.56	1.22
<i>substitute(50)</i>	Pa.	0	0	0.6	0.56	1.32
<i>substitute(60)</i>	Pa.	0	0	0.6	0.55	1.39
<i>substitute(70)</i>	Pa.	0	0	0.6	0.55	1.47
<i>substitute(80)</i>	Pa.	0	0	0.6	0.55	1.51
<i>substitute(90)</i>	Pa.	0	0	0.6	0.54	1.55
<i>substitute(100)</i>	Pa.	0	0	0.6	0.54	1.59
<i>substitute(110)</i>	Pa.	0	0	0.6	0.54	1.62
<i>substitute(120)</i>	Pa.	0	0	0.6	0.54	1.64
<i>substitute(200)</i>	Pa.	0	0	0.6	0.54	1.77
<i>fib(1)</i>	Se.	0	0	0.53	0.53	1.0
<i>fib(2)</i>	Se.	0	0	0.6	0.59	0.64
<i>fib(3)</i>	Se.	0	0	0.6	0.56	0.82
<i>fib(4)</i>	Pa.	0	0	0.6	0.53	1.04
<i>fib(5)</i>	Pa.	1	0	0.6	0.52	1.0
<i>fib(6)</i>	Pa.	1	0	0.6	0.51	1.0
<i>fib(7)</i>	Pa.	1	0	0.6	0.51	1.0
<i>fib(8)</i>	Pa.	1	0	0.6	0.51	1.0
<i>fib(9)</i>	Pa.	1	0	0.6	0.5	1.0
<i>fib(10)</i>	Pa.	1	0	0.6	0.5	1.0
<i>fib(11)</i>	Pa.	1	0	0.6	0.5	1.0
<i>fib(12)</i>	Pa.	1	0	0.6	0.5	1.0
<i>fib(13)</i>	Pa.	1	0	0.6	0.5	1.0
<i>fib(14)</i>	Pa.	1	0	0.6	0.5	1.0
<i>hanoi(1)</i>	Se.	0	0	0.53	0.53	1.0
<i>hanoi(2)</i>	Se.	0	0	0.6	1	1.0
<i>hanoi(3)</i>	Se.	0	0	0.6	0.9	1.0
<i>hanoi(4)</i>	Se.	0	0	0.6	0.68	1.0
<i>hanoi(5)</i>	Se.	0	0	0.6	0.58	0.94
<i>hanoi(6)</i>	Pa.	0	0	0.6	0.53	1.28
<i>hanoi(7)</i>	Pa.	1	0	0.6	0.51	1.0
<i>hanoi(8)</i>	Pa.	1	0	0.6	0.5	1.0
<i>hanoi(9)</i>	Pa.	1	0	0.6	0.5	1.0
<i>hanoi(10)</i>	Pa.	1	0	0.6	0.5	1.0
<i>hanoi(11)</i>	Pa.	1	0	0.6	0.5	1.0
<i>hanoi(12)</i>	Pa.	1	0	0.6	0.5	1.0
<i>hanoi(13)</i>	Pa.	1	0	0.6	0.5	1.0
<i>hanoi(14)</i>	Pa.	1	0	0.6	0.5	1.0

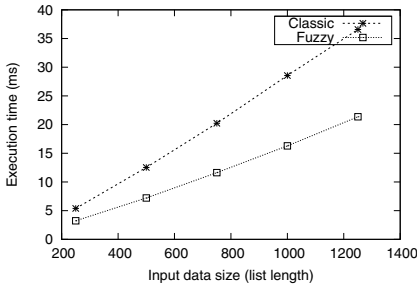


Fig. 9. Qsort selected executions

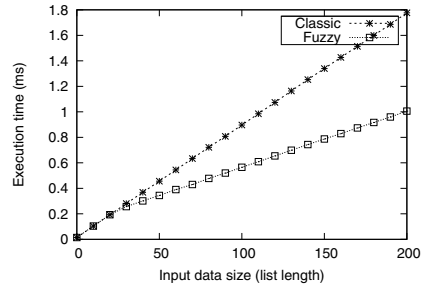


Fig. 10. Substitute selected executions

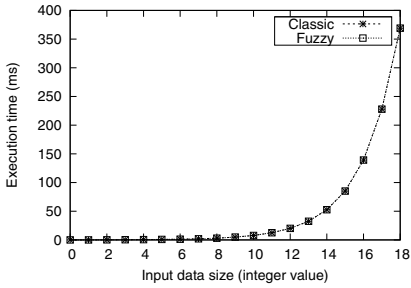


Fig. 11. Fibonacci selected executions

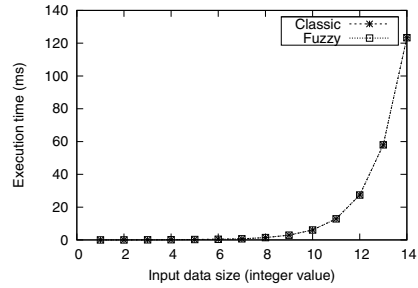


Fig. 12. Hanoi selected executions

Table 5 shows the experimental results. The first four columns show the same information as in Tables 2,3, although in Table 5 the first column refers to real benchmarks, namely: `qsort(n)` sorts a list of n random elements; `substitute(n)` replaces by 2 the x 's that appears in an expression of the form $(x + x \dots)$ composed by n '+'s and $n + 1$ x 's; `fib(n)` obtains the n th Fibonacci number; and `hanoi(n)` solves the Towers of Hanoi problem with 3 rods and n disks. P_c , S_c , P_2 and S_2 are also defined as in Tables 2,3. Note that in this case, in order to determine the optimal execution, both sequential and parallel execution times have been measured directly over the platform. The last column shows the `speedup` of our fuzzy approach with respect to the conservative approach: $speedup = \frac{T_c}{T_f}$, where T_c is the time of the selected execution using the conservative approach and T_f is the time of the selected execution using our fuzzy approach. A value bigger than one of `speedup` means that the execution selected with our approach is faster than the one selected by the conservative one.

We can distinguish two main sets of cases in Table 5: on one hand `qsort` and `substitute`, and on the other hand `fib` and `hanoi`. In the first set the *upper bound* on the sequential execution time is different from the *lower bound* whereas in the second set, both bounds are equal. This is understandable, since the execution time for the first set of cases not only depends on the length of the input list, but also on the values of its elements. Thus, for a given list length, there may be different execution times, depending on the actual values of the lists with such length. However, in the second set of cases, the execution time only depends on the size (using the integer value metric) of the input argument, and all executions for the same input data size take the same execution time.

Our approach provides better average case behavior than the conservative approach in both cases.

Figures 9, 10, 11 and 12 show, in detail, how both approaches work in particular cases in a graphical way. Input has the same meaning that in previous Table 5 and execution times are presented in milliseconds. In all the figures the conservative approach is called `Classic` and represented with a stars line while our approach is called `Fuzzy` and its symbol is a white square.

Figures 11 and 12 show how both approaches have nearly the same behavior for all the tested cases for `fibonacci` and `hanoi`. In fact, at this scale, the scarce cases in which there is a slowdown (see Table 5) cannot be appreciated. Figure 9 show the behavior for `qsort` and `substitute`. It is clear how the times of the executions selected by our approach are smaller (except in a small number of cases that is insignificant), and how the difference between both approaches becomes bigger when input data sizes increase.

6 Conclusions

We have applied fuzzy logic to the program optimization field, in particular, to automatic granularity control in parallel/distributed computing. We have derived fuzzy conditions for deciding whether to execute some tasks in parallel or sequentially, using information about the cost of tasks and parallel execution overheads.

We have performed an experimental assessment of the fuzzy conditions and identified the ones that have the best average case behavior. We have also compared our proposed fuzzy conditions with existing sufficient (conservative) ones for performing granularity control. Our experiments showed that the proposed fuzzy conditions result in better program optimizations (on average) than the conservative conditions. The conservative approach ensures that execution decisions will never result in a slowdown, but loses some parallelizations opportunities (and thus, no speedup is obtained). In contrast, the fuzzy approach makes a better use of the parallel resources and although fuzzy conditions can produce slowdown for some executions, the whole computation benefits from some speedup on average (always preserving correctness). Of course, the fuzzy approach is applicable in scenarios where the no slowdown property is not needed, as for example video games, text processors, compilers, etc.

Experiments performed with real programs (and real execution times) have demonstrated that our approach can be successfully applied in practice. We intend to perform a more rigorous and broad assessment of our approach, by applying it to larger real life programs and using fully automatic tools for estimating execution times.

Although a lot of work still remains to be done, the preliminary results are very encouraging and we believe that it is possible to exploit all the potential offered by multicore systems by applying fuzzy logic to automatic resource aware program parallelization techniques.

Acknowledgements. This research has been partially funded by the EU 7th. FP NoE *S-Cube* 215483, FET IST-231620 *HATS*, MICINN TIN-2008-05624 *DOVES* and CM project P2009/TIC/1465 *PROMETIDOS*. Teresa Trigo has been supported by CAM grant CPI/0621/2008.

References

1. Baldwin, J.F., Martin, T., Pilsworth, B.: *Fril: Fuzzy and Evidential Reasoning in Artificial Intelligence*. John Wiley & Sons (1995)
2. Debray, S.K., Lin, N.W., Hermenegildo, M.: Task Granularity Analysis in Logic Programs. In: *Proc. of the 1990 ACM Conf. on Programming Language Design and Implementation*, pp. 174–188. ACM Press (June 1990)
3. Guadarrama, S., Muñoz, S., Vaucheret, C.: Fuzzy Prolog: A new Approach Using Soft Constraints Propagation. *Fuzzy Sets and Systems* FSS144(1), 127–150 (2004) iSSN 0165-0114
4. Hermenegildo, M., Puebla, G., Bueno, F., López-García, P.: Integrated Program Debugging, Verification, and Optimization Using Abstract Interpretation (and The Ciao System Preprocessor). *Science of Computer Programming* 58(1–2) (2005)
5. Hermenegildo, M., Rossi, F.: Strict and Non-Strict Independent And-Parallelism in Logic Programs: Correctness, Efficiency, and Compile-Time Conditions. *Journal of Logic Programming* 22(1), 1–45 (1995)
6. Hermenegildo, M.V., Bueno, F., Carro, M., López, P., Morales, J.F., Puebla, G.: An Overview of the Ciao Multiparadigm Language and Program Development Environment and its Design Philosophy. In: Degano, P., De Nicola, R., Bevilacqua, V. (eds.) *Concurrency, Graphs and Models*. LNCS, vol. 5065, pp. 209–237. Springer, Heidelberg (2008)
7. Huelsbergen, L., Larus, J.R., Aiken, A.: Using Run-Time List Sizes to Guide Parallel Thread Creation. In: *Proc. ACM Conf. on Lisp and Functional Programming* (June 1994)
8. Ishizuka, M., Kanai, N.: Prolog-ELF incorporating fuzzy logic. In: *IJCAI*, pp. 701–703 (1985)
9. Kruatrachue, B., Lewis, T.: Grain Size Determination for Parallel Processing. *IEEE Software* (January 1988)
10. Lee, R.: Fuzzy logic and the resolution principle. *Journal of the Association for Computing Machinery* 19(1), 119–129 (1972)
11. Li, D., Liu, D.: *A Fuzzy Prolog Database System*. John Wiley & Sons, New York (1990)
12. López-García, P., Hermenegildo, M., Debray, S.K.: A Methodology for Granularity Based Control of Parallelism in Logic Programs. *Journal of Symbolic Computation, Special Issue on Parallel Symbolic Computation* 21(4–6), 715–734 (1996)
13. McGreary, C., Gill, H.: Automatic Determination of Grain Size for Efficient Parallel Processing. *Communications of the ACM* 32 (1989)
14. Mera, E., López-García, P., Carro, M., Hermenegildo, M.: Towards Execution Time Estimation in Abstract Machine-Based Languages. In: *10th Int'l. ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP 2008)*, pp. 174–184. ACM Press (July 2008)
15. Pablos-Ceruelo, V., Strass, H., Muñoz Hernández, S.: Rfuzzy—a framework for multi-adjoint fuzzy logic programming. In: *Annual Meeting of the North American of Fuzzy Information Processing Society, NAFIPS 2009*, pp. 1–6 (June 2009)
16. Zhong, X., Tick, E., Duvvuru, S., Hansen, L., Sastry, A., Sundararajan, R.: Towards an Efficient Compile-Time Granularity Analysis Algorithm. In: *Proc. of the 1992 International Conference on Fifth Generation Computer Systems, Institute for New Generation Computer Technology (ICOT)*, pp. 809–816 (June 1992)

Fuzzy and Fractal Technology in Market Analysis

Petr Kroha and Marcus Lauschke

Department of Computer Science, University of Technology,
Strasse der Nationen 62, 09111 Chemnitz, Germany

kroha@informatik.tu-chemnitz.de, m.lauschke@gmx.de

<http://www.tu-chemnitz.de/informatik/ISST>

Abstract. In this contribution, we describe our investigation of using fuzzy and fractal technology for analysing time series of market data. For having a comparison, we implemented the commonly accepted technical indicator method. Then, we implemented and tested a fuzzy component that provides fuzzyfication by the Mamdani Larsen inference method with static rules using not only Gauss but also Cauchy and Mandelbrot distribution. In the sequel, we implemented and tested a fractal component that provides fuzzy clustering by the Takagi Sugeno method with dynamic fuzzy rules. Looking for an optimum, we simulated many parameter combinations and compared the results. We compared the results obtained and present some interesting results of our experiments.

Keywords: Time series of market data, Technical indicators, Fuzzy controller, Static fuzzy rules, Mamdani Larsen method, Dynamic fuzzy rules, Takagi Sugeno method.

1 Introduction

Currently, it is not difficult to collect and store very large data representing time series. However, there is a question whether it is possible to extract any information usable for trend forecasting (meteorology, biology, seismology, finance) and how to do it.

Data about financial markets is very interesting. There are large time series available and the eventually obtained forecast can be easily tested. The question is whether a forecast exists, of course.

There are different hypotheses about processes in markets. Under Efficient market hypothesis [6], [18], markets were assumed to be efficient in the sense that prices reflected all current information that could anticipate future events. There is a statistical requirement that market returns were normally distributed as white noise. This traditional capital market theory has been modeled by probabilities since the first approach in [1] (originally published in 1900 as Ph.D. Thesis).

More recently, Markowitz [19], [20] used the standard deviation as a measure of the risk of investment, and the covariance of returns as a measure of diversification of investment, where uncorrelated or negative correlated stocks reduced the risk of portfolio. The next famous work based on probabilities is Black-Scholes option pricing model [2].

Later, some anomalies in market development have been found. They were explained by the fact that different investors have different access to information (e.g. insiders

know more), different investment horizon (short-term investors, long-term investors), and different interpretation procedures. Based on these phenomena, an Inefficient market hypothesis was formulated [24] but the anomalies cannot be modeled easily.

The probability model of markets used in [6], [19], [2] has one advantage and one disadvantage:

- The advantage is that it can be simply described by tools for Gaussian statistics.
- The disadvantage is that the measured data, i.e. the market returns, are not distributed normally. Using the current computer technology and the known time series (e.g. 103-years known daily prices of Dow Jones Industrial), the difference between the theoretically supposed distribution and the distribution found by experiments is very significant. This was documented in many works starting with Mandelbrot [17], [21] and others. Compared to normal distribution, the real distribution of market returns is characterized by asymmetry, by higher peaks at the mean and fatter tails that do not converge to zero.

The Fractal market hypothesis [21], [22] places no statistical requirements on the market development process. The goal is to include the investor behavior and to find a model that fits to observed time series. The components of investor behavior are investment horizon and crowd behavior. Investors have different horizons of investment. Hence, they have different strategies for buying and selling. Further, there are panics and stampedes caused by the known crowd behavior of investors. It has been found that markets have a memory and their behavior is not characterized by white noise (no memory) as suggested in [6] but by black noise [21].

All investors are interested in the prediction of market movements. Short-term investors follow technical analysis, long-term investors follow fundamental analysis. Often a combination of both approaches will be used.

Theoretically, market movements cannot be predicted successfully as both efficient market theory and fractal market theory say. But there is not only local noise. Also, non-regular and non-periodic global deterministic movements called “trends” are present. It is very probably not predictable when they start and how long they take. Investors try to estimate the trend begin and the trend end and use loss-limiting strategies that control buying and selling stocks.

In this paper, we do not discuss the possibility that the market can be intentionally influenced by big purchases or sells. However, it is known that large investments can be moved only to start an artificial trend and to use it.

So, the goal of such a strategy is to indicate that a trend started, resp. finished and generate a corresponding buy signal, resp. a sell signal, in such a way that the gain is greater than the loss in the investment horizon. To simplify the problem, we do not discuss problems of taxes, problems of money management, problems of hedging and other more complex strategies that are used by traders and investors. More or less, these aspects are parts of the market parameters that change their weighting chaotically corresponding to the changes in regulations, to the behavior and sentiment of investors.

The motivation of our project is to build controllers based on fuzzy and fractal technology and test, what can be gained with fuzzy and fractal strategies compared to the often used strategies based on technical indicators of technical analysis.

Our original approach is that we implemented and tested not only Gauss but also Cauchy and Mandelbrot distributions in our fuzzy component. The published fuzzy-controllers discussed in Section 2 (Related work) use Gaussian distribution of price deviations even though it is known that the Gaussian normal distribution does not fit to the reality very well. We compared the results and found that the Cauchy distribution is more efficient than the others.

Further, we implemented and tested a fractal component using fuzzy clustering and the Takagi Sugeno method of dynamic fuzzy rules. This method brings the best results. All methods were tested on daily prices of 100 stocks of NASDAQ100 (see Section 9 for more details).

The rest of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we introduce the developed system. In Sections 4, 5, and 6, its components are described. The synthesis of the components is discussed in Section 7. The goals of our investigation are explicitly stated in Section 8. Section 9 describes the implementation, experiments, and results. In the last section we conclude.

2 Related Work

The idea that the market returns are not normally distributed is not new. It has been published in [17]. In [5], a system using the Mamdani Larsen fuzzy inference method is described but with the Gaussian distribution in background.

In [3], a system using the Takagi Sugeno inference method was used to calculate the fractal dimension of a time series. We extended this approach using the Hurst exponent and correlation quotient.

In [23], a system with technical indicators and lately a fuzzy system (using triangular membership function) with static fuzzy rules is described. This system was tested with data from January, 1st, 2000 to July, 7th, 2006 and documented that the fuzzy system performed better than the system using technical indicators. In our system, we used dynamic fuzzy rules and non-Gaussian distribution for membership function.

3 The System Developed

The main components of our system and their features:

- technical indicator component
(used MA, TBI, MACD, MAcut, dTD)
- fuzzy-control component
(ROC, stochastic indicator, and support/resistance indicator are used to get input data for fuzzy component),
 - static fuzzy rules,
 - Mamdani Larsen fuzzy inference,
 - defuzzification.
- fractal analysis component using additional input data
(fractal dimension, Hurst exponent, correlation, trend)
 - fuzzy clustering and dynamic fuzzy rules (Takagi Sugeno inference method),
 - interpretation of the result.

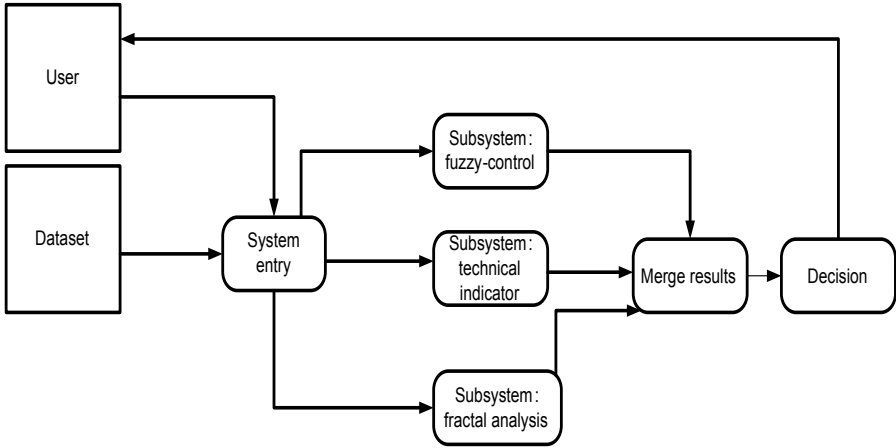


Fig. 1. The system architecture and dataflow

- decision strategies,
- synthesis of results.

These parts will be discussed in the next sections.

4 Technical Indicators Component

Technical analysis is based on a study of patterns on charts, as well as price trends, as well as support and resistance levels at which rising or falling trends may be halted or reversed.

The main idea is that all information necessary to forecast the market is stored in the existing time series. Some technical indicators have been defined to indicate a trend’s begin, its end, and its strength. Most short-term investors use technical analysis because it reflects the current investors’ behavior.

The effectivity of technical indicators has been investigated by [7], [8].

We have used it for two purposes:

- First, to get input parameters for the fuzzy control component (technical indicators [26] used: rate of change indicator, stochastic indicator, and Support/resistance indicator)—see Section 5.
- Second, to get input parameters for the technical indicator component (technical indicator used: MA, TBI, MACD, MAcut, dTD)—see Section 9.2. We normalized the used technical parameters into the interval [0..1].

5 Fuzzy Component

Fuzzyfication provides the transformation of numeric input data (sharp data) into fuzzy data (unsharp data). Values of technical indicators such as rate of change indicator, stochastic indicator, and Support/resistance indicator have been used as input data.

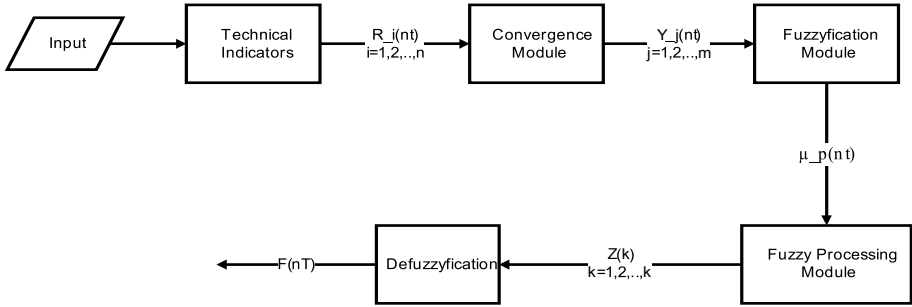


Fig. 2. Fuzzy component

5.1 Technical Indicators as Input for the Fuzzy Component

In this component, technical indicators will be computed to be used as basics of input data for the fuzzy component.

Rate of Change(ROC). This indicator describes the absolute difference between the the current stock price and the price *n* days ago:

$$\begin{aligned}
 sp &= \text{last closing stock price} \\
 ROC &= sp(\text{today}) - sp(\text{today} - \text{ndays})
 \end{aligned}$$

Stochastic indicator. The main idea behind stochastic indicator is that rising price tends to close near its previous highs, and falling price tends to close near its previous lows (definitions are given below). K - D stochastic indicatr was introduced by Lane [15]. Usually, indicator *K*(*nT*) (denoted often as %K - fast line) and *D*(*nT*) (denoted often as %D - slow line) are used. We computed two values for time interval *n*, where

$$\begin{aligned}
 sp &= \text{last closing stock price} \\
 lp &= \text{the lowest price} \\
 hp &= \text{the highest price} \\
 ap &= \text{average price of m days} \\
 K(nT) &= \frac{sp(\text{today}) - lp}{hp - lp} * 100 \\
 D(nT) &= \sum_{i=n-3}^n \frac{K(iT)}{3}; n \geq 3
 \end{aligned}$$

Low resp. high price means here the lowest resp. the highest stock price in the given time interval. The lag of 3 days used in *D*(*nT*) is a value recommended by traders. Very probably, it represents an experience that price changes older that 3 days have a very small influence.

Support/Resistance Indicator

sl = Support level

rl = Resistance level

$$sl = Avg(nT) - 2 * \sigma(nT),$$

$$rl = Avg(nT) + 2 * \sigma(nT),$$

where

$$\sigma(nT) = \sqrt{\frac{\sum_{i=n-m}^n (sp(day_i) - Avg(day_i))^2}{m}},$$

$$Avg(nT) = \frac{\sum_{i=n-m}^n sp(day_i)}{m}$$

5.2 Convergence Module - More Input Parameters

In this component, the indicators mentioned above are used to generate more parameters. We used the following equations from [5]:

$$Y_{ROC}(nT) = \frac{R(nT) - R((n - 30)T)}{R((n - 30)T)}, n \geq 30,$$

$$Y_{d(ROC)}(nT) = Y_{ROC}((n - 2)T) - Y_{ROC}(nT), n \geq 2,$$

$$Y_D(nT) = D(nT),$$

$$Y_K(nT) = K(nT),$$

$$Y_{D-K}(nT) = Y_D(nT) - Y_K(nT),$$

$$Y_{Res}(nT) = Avg(nT) + 2 * \sigma(nT) - R(nT), n \geq 30,$$

$$Y_{Sup}(nT) = R(nT) - (Avg(nT) + 2 * \sigma(nT)), n \geq 30,$$

$$Y_{Avg}(nT) = R(nT) - Avg(nT), n \geq 30,$$

$R(nT)$ is the stock price on the n-th day, $D(nT)$ und $K(nT)$ are indicators defined above and $Avg(nT)$ is the average stock price during the observation time interval.

5.3 Fuzzyfication, Fuzzy Processing, and Defuzzyfication

This part of our system generates forecast using the Mamdani Larsen inference method. The membership functions and rules are implemented as being static. The indicators described above have been used as input parameters in a similar way as in [5].

We used 11 fuzzy rules for fuzzyfication and the Gaussian bell function (in the first approach - the improvement is given in Section 9.1) as the membership function ($SUP = 100$ und $INF = 0$). The output membership function is shown in Fig. 3.

The output membership function is sent to the defuzzyfication module. We used the center-of-area method producing a value in the interval [0..100].

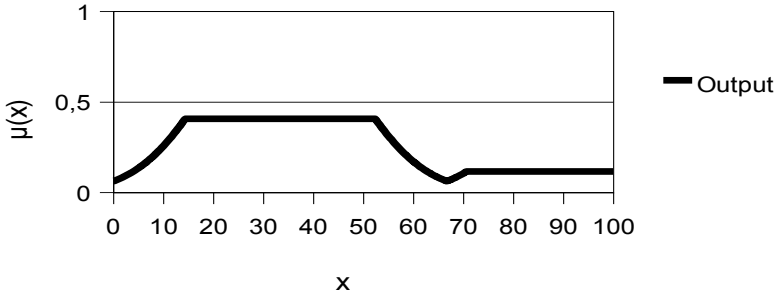


Fig. 3. Output membership function

6 Fractal Analysis Component

This component uses stock prices as input data. In the part Fractal data calculation (see Fig. 4), the following data will be computed:

- box dimension and fractal dimension,
- Hurst exponent,
- correlation,
- trend in interval.

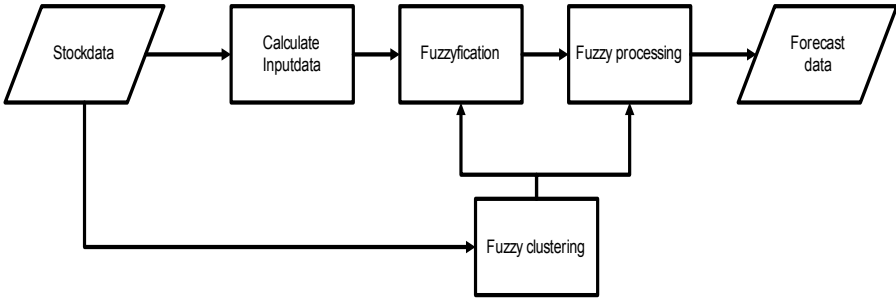


Fig. 4. Fractal architecture

The fractal dimension of the time series is a real number of the interval [1;2] which can be seen as a metric measuring how much the time series is jagged. A value near 1 means that there is a trend similar to a line, a value near 2 means that there are very many positive or negative changes in the interval. To measure the fractal dimension, a box covering method is used. The graph of the time series has to be covered by a set of smallest quadratic nonoverlapping boxes of the same size. The fractal dimension (exact: the fractal capacity dimension) is the number of such boxes that contain at least one point of the object [3]. Then, we can calculate the fractal dimension using the following formula:

$$Dim = \frac{\log_{10}(number\ of\ boxes)}{\log_{10}(\frac{1}{size\ of\ boxes})} \tag{1}$$

The box size is normalized in relation to the size of the interval.

To obtain the Hurst exponent, we first have to eliminate all linear trends from the data. Then, we define a time interval N and find the range R and the standard deviation S in this interval. We can derive the Hurst quotient as $\frac{R}{S}$ or calculate the Hurst exponent (values in interval $[0;1]$) as:

$$H = \frac{\log(\frac{R}{S})}{\log(\frac{N}{2})} \tag{2}$$

Time series with H near to 1 have trends, time series with H near to 0 are near to a white noise and no trend can be found and forecasted. The Hurst exponent has a close relation to the fractal dimension:

$$Dim = 2 - H \tag{3}$$

The correlation quotient (value of interval $[-1;1]$) specifies the linear correlation between elements of the time series. A value near to 1 indicates a positive trend, values near to -1 indicate a negative trend. The following formula (Bravais-Pearson) will be used (\bar{x} is a mean value):

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} \tag{4}$$

The trend in an interval is calculated as the difference between the stock value at the begin and at the end of the interval.

The fuzzy-component used for the fractal analysis subsystem (Fig. 1) uses fractal data (fractal dimension, Hurst exponent, correlation coefficient, and trend) as input parameters. They are processed by a fuzzy control system using the Takagi Sugeno inference method. The membership function and the rules are not defined as being static. They will be found dynamically using fuzzy clustering of time series in a similar way as in [3].

We assigned the cluster number to 4, being inspired by (low, medium, big, large). The clustering is analysed by the c-mean clustering method. It gives us the weights a_{ij} we need for the output function calculation. In our system, this process will be repeated any 150 days with the last 364 data sets.

Having the membership functions given by clustering the fuzzyfication can start to calculate the output function directly according to:

$$y_{res} = \frac{\sum_{i=1}^n E_i * (a_{i0} + a_{i1} * x_1 + \dots + a_{im} * x_m)}{\sum_{i=1}^n E_i} \tag{5}$$

There are more methods available how to calculate variables E_i which denote how much the rule R_i matches. We have used c-mean clustering that calculates E_i using

$$E_i = \frac{1}{(\sum_{k=1}^c \|\mathbf{x} - \mathbf{c}_k\|)^{\frac{2}{m-1}}} \tag{6}$$

The value of the output function has to be qualified in some way, i.e. we have to specify the threshold for buy and sell signals. To find some empirical values we analysed 100

stocks in the time interval 2003 – 2009. We found using simulation that it was most effective (at least in the analysed time interval) to have +6% as a threshold that should be crossed to generate a buy signal and –6% as a threshold for sell signal.

7 Synthesis of Component Results

In the previous parts, all three components have been described. The obtained results, i.e. the buy and sell signals, have to be merged together as shown in Fig. 11

First, we used the following merging formula, which we denote as absolute merging:

$$result = rFC + \frac{(rTI - 0.5) * 100}{2} + \frac{rFR - 50}{2} \tag{7}$$

where rFC is the result of fuzzy-control, rTI is the result of the technical indicators, rFR is the result of the fractal component. The variable $result$ is in the interval $[-50...150]$.

Second, we used the following formula, which can be denoted as mean merging:

$$FC = doFC * rFC \tag{8}$$

$$FI = doTI * (rTI * 100) \tag{9}$$

$$FR = doFR * rFR \tag{10}$$

$$result = \frac{FC + FI + FR}{doFC + doTI + doFR} \tag{11}$$

where variables $doFC$, $doTI$, and $doFR$ have values 0 or 1 and indicate whether the results of the fuzzy control (FC), the technical indicators (TI), and/or the fractal control (FR) are present.

7.1 Decision Strategies

The resulting value of the whole system is a numeric one, but we need a qualitative value for the decision. This means, we need thresholds for the definition of signals for buy and sell. We defined two thresholds UTL (upper limit) and LTL (lower limit) and implemented the following strategies:

- Low risk strategy - $LTL = 49$ and $UTL = 51$ - very careful but the frequency of buy and sell can be very high.
- High risk strategy - $LTL = 40$ and $UTL = 60$ - more risk but the frequency of buying and selling is not as high.
- Variable border - the last values (0 means all past data, 256 means the last 256 days) have been analysed and the best combination of LTL and UTL will be used.

In Section 9 we show which decision strategy brings the best results.

8 Goals of Our Investigation

As we stated in Section 1, market returns (exactly, increments of market returns) are not distributed normally in real markets, even though normal distribution will be used in routine business. We stated the following questions as objectives of our investigation:

- which parameter combination used in the technical indicator component will bring the highest gain,
- which distribution function will bring the highest gain when used in the fuzzy-controller,
- which tuple of fractal analysis parameters will bring the highest gain,
- which component of our system will generate the highest gain,
- which combination of components can generate more gain than each of them separately.

9 Implementation, Experiments, and Results

The presented system has been implemented in Java as a multi-threaded component of our information system [9], [10], [11], [12], [13], [14]. A detailed description of the design and implementation of the presented system is out of the scope of this paper and is given completely in [16].

For our experiments with the implemented system we used time series of all stocks from NASDAQ100 (daily prices) in the time interval from January, 1st, 2003 to October, 1st, 2009. When looking for the parameters' value by simulation, we used an investment of \$ 10.000 and transaction costs of \$ 10. The implemented system runs on Apple MacPro with 8 cores, 8 GB of memory, and 2.26 GHz.

9.1 Non-gaussian Distribution Used in the Fuzzy Component Contribution

In the following Table 1, we can see the results, i.e., the value (in thousands of the invested \$ 10,000) when changing the distribution in the fuzzy-controller and the strategy. The elapsed time was about 15 minutes.

Table 1. Results of Fuzzy-Control component

Strategy	Gauss	Cauchy	Mandelbrot
Low Risk	\$ 15.3	\$ 16.8	\$ 14.5
High Risk	\$ 17.8	\$ 19.9	\$ 19.3
Var. Border 0	\$ 17.7	\$ 19.7	\$ 18.3
Var. Border 256	\$ 15.2	\$ 14.7	\$ 15.1

We found that in most cases (Table 1) the Cauchy distribution used in the fuzzy-controller gives the best results (about 99 %) and the Gauss distribution the worst results (about 78 %). Considering strategies, the high risk strategy results were the best.

9.2 Technical Indicators Component and Its Contribution

We used the technical indicators with the following parameters: Moving Average (17), TBI (9,17), TBI-line = 100, MACD(12,26) - always calculated any 3 days. The highest gain (about 60%) was achieved for the most simple merging (for variable border 0), but it was rather small compared to fuzzy-control. Hence, we do not describe the details about the used methods of technical indicators merging here.

9.3 Fractal Component and Its Contribution

In this experiment, we used the 6 % threshold as explained above and used the rules recomputation, i.e. the new clustering, every 150 days. The highest gain (about 100 %) was achieved when using the tuple [box-dimension;correlation] for fuzzy clustering. The gain was higher than that of the other components. To get the results, we needed about 32 minutes.

Table 2. Results of fractal component

Input	Result
Box-corr	\$ 20,024.19
Hurst-corr	\$ 17,797.11
Box-Hurst-corr	\$ 16,038.22
Box-Hurst-corr-Trend	\$ 16,951.81

9.4 Synthesis of Components and Their Contributions

We just described the individual behavior of components. The next question was whether we can get more profit when using some specific combination of the components' results. Because of the very large number of possible combinations, the simulation altogether took about 71 hours. The best gain of 88.4 % was obtained with the following parameters of our system:

- synthesis of the components' results = mean,
- distribution used in fuzzy-controller fuzzyfication = Cauchy,
- clustering in fractal analysis according to [Hurst exponent; correlation],
- decision strategy = high risk.

We do not discuss the details of the synthesis because we can see that the best solution is to use only the fractal component (higher gain than the combination with other components).

9.5 Comparison with the Strategy Buy and Hold

One of the often used strategies is Buy & Hold. A stock will be bought at the begin of the interval and sold at the end of interval. In the next experiment, we choose an interval (21.11.2000 - 26.9.2008) in which the value of the German market index DAX was more or less equal at the begin and at the end.

Table 3. Comparison with Buy & Hold - Fuzzy methods, Fractal methods, and Technical indicators

Index	Buy & Hold	Fuzzy	Fractal	Techn. Ind.	The best combination
DAX	0.9365	1.2236	1.0326	1.5593	0.6884
Nasdaq100	0.6636	0.5206	0.8755	0.6306	0.7256

Then, we simulated how much an investor would have earned when using our fuzzy or fractal controller for his/her transactions. The Table 3 contains quotients that denote how many times the investement would have been increased. We can see that the fractal method brings slightly better results than the Buy & Hold strategy but in this case technical indicators bring good results.

The number of transactions is important, too. The Buy & Hold used only 2 transactions, fuzzy approach used 7 transactions, technical indicators 70 transactions, and fractal approach 117 transactions.

10 Conclusions

As we have discussed, market returns are not distributed normally and the Gauss distribution used in our fuzzy-controller delivered the worst results. According to the Fractal market hypothesis, non-periodical trends exist that correspond to the global determinism component of the process and black noise that corresponds to the local randomness component.

The result is that it is not possible to forecast market changes but it seems to be possible to achieve some gain in the long-term investment when using fuzzy or fractal technology. Of course, we did not consider taxes but they are different in various countries and changes in time. Further, the data processing, especially in the case of fractal analysis, is very time consuming as given above.

In the light of the recent financial crisis, we have to mention that we can only use public data for our processing and forecasting. May be that there is a currently hidden accounting fraud by a company like by Enron in 2001 which was named by the magazine Fortune as the America's Most Innovative Company for six consecutive years or by Lehman & Brothers in 2008. May be that there is a currently hidden accounting fraud by a state caused by years of unrestrained spending like in Argentina in 1999 or in Greece in 2008. We cannot expect too much from results of public data processing because we do not know anything about the data quality.

In further work, we will try to combine the fuzzy and fractal methods with our text classification of market news [11], [12], [13], [14]. The goal would be to investigate whether the result could be improved.

References

1. Bachelier, L.: Theory of speculation. In: Cootner, P. (ed.) The Random Character of Stock Market Prices, M.I.T. Press, Cambridge (1964) (Originally published in 1900 as Ph.D. Thesis)

2. Black, F., Scholes, M.: The Pricing of Options and Corporate Liabilities. *Journal of Political Economy* (May/June 1973)
3. Castillo, O., Melin, P.: Hybrid Intelligent Systems for Time Series Prediction Using Neural Networks, Fuzzy Logic, and Fractal Theory. *IEEE Transactions on Neural Networks* 13(6) (2002)
4. Castillo, O., Melin, P.: Evolutionary design and applications of hybrid intelligent systems. *Int. J. Innovative Computing and Applications* 1(1) (2007)
5. Dourra, H., Siy, P.: Investment using technical analysis and fuzzy logic. *Fuzzy Sets and Systems* 127(2), 221–240 (2002)
6. Fama, E.: Efficient capital markets: A review of theory and empirical work. *Journal of Finance* 25, 383–417 (1970)
7. Hellstroem, T., Holmstroem, K.: Predicting the Stock Market. Technical Report Series IMA-TOM-1997-07. Maelarden University (1997)
8. Hellstroem, T., Holmstroem, K.: Predictable Patterns in Stock Returns. Technical Report Series IMA-TOM-1997-0. Maelarden University (1997)
9. Kroha, P., Gemeinhardt, L.: Using XML in a Web-oriented Information System. In: Tjoa, A.M., Wagner, R.R. (eds.) *Proceedings DEXA 2001, Workshop Network-Based Information Systems, 12th International Workshop on Database and Expert Systems Applications*, pp. 217–221 (2001)
10. Kroha, P., Baeza-Yates, R.: A Case Study: News Classification Based on Term Frequency. In: *Proceedings of 16th International Conference DEXA 2005, Workshop on Theory and Applications of Knowledge Management TAKMA 2005*, pp. 428–432 (2005)
11. Kroha, P., Baeza-Yates, R., Krellner, B.: Text Mining of Business News for Forecasting. In: *Proceedings of 17th International Conference DEXA 2006, Workshop on Theory and Applications of Knowledge Management TAKMA 2006*, pp. 171–175 (2006)
12. Kroha, P., Reichel, T.: Using Grammars for Text Classification. In: Cardoso, J., Cordeiro, J., Filipe, J. (eds.) *Proceedings of the 9th International Conference on Enterprise Information Systems ICEIS'2007, Volume Artificial Intelligence and Decision Support Systems*, pp. 259–264 (2007)
13. Kroha, P., Reichel, T., Krellner, B.: Text Mining for Indication of Changes in Long-Term Market Trends. In: Tochtermann, K., Maurer, H. (eds.) *Proceedings of I-KNOW 2007 7th International Conference on Knowledge Management as part of TRIPLE-I 2007, Journal of Universal Computer Science*, pp. 424–431 (2007)
14. Kroha, P., Nienhold, R.: Classification of Market News and Prediction of Market Trends. In: *ICEIS 2010. Artificial Intelligence and Decision Support Systems*, vol. 2, pp. 187–192 (2011)
15. Lane, G.: Lane's Stochastics. *Technical Analysis of Stocks and Commodities magazine*, 2nd edn., pp. 87–90 (May/June 1984)
16. Lauschke, M.: Time series analysis of stock prices using fuzzy logic, technical indicators, fractal analysis. University of Technology Chemnitz, M.Sc. Thesis (2010) (in German)
17. Mandelbrot, B.: The variation of certain speculation prices. IBM Research Report (1962)
18. Malkiel, B.: *A Random Walk Down Wall Street*. W.W. Norton, New York (1996)
19. Markowitz, H.: Portfolio Selection. *Journal of Finance* 7 (1952)
20. Markowitz, H.: *Portfolio Selection: Efficient Diversification of Investments*. John Wiley (1959)
21. Peters, E.: *Fractal market analysis*. John Wiley (1994)
22. Peters, E.: *Chaos and Order in the Capital Markets*, 2nd edn. John Wiley (1996)
23. Raimondi, F., Via, P., Mulè, M.: A New Fuzzy Logic Controller for Trading on the Stock Market. In: *Proceedings of Conference ICEIS*, pp. 322–329 (2007)

24. Shleifer, A.: *Inefficient Markets – An Introduction to Behavioral Finance*. Oxford University Press (2000)
25. Setnes, M., van Drempt, O.: *Fuzzy modeling in stock-market analysis*. Delft University of Technology (2001)
26. Kirkpatrick, C., Dahlquist, J.: *Technical Analysis: The Complete Resource for Financial Market Technicians*. Financial Times Prent. Int. (2006)

The Banach Contraction Principle in Fuzzy Quasi-metric Spaces and in Product Complexity Spaces: Two Approaches to Study the Cost of Algorithms with a Finite System of Recurrence Equations

Francisco Castro-Company, Salvador Romaguera, and Pedro Tirado*

Instituto Universitario de Matemática Pura y Aplicada, Universidad Politécnica de Valencia

Camino de Vera s/n, 46022 Valencia, Spain

{fracasco, sromague, pedtipe}@mat.upv.es

<http://www.iumpa.upv.es>

Abstract. Considering recursiveness as a unifying theory for algorithm related problems, we take advantage of algorithms formulation in terms of recurrence equations to show the existence and uniqueness of solution for the recurrence equations associated to a kind of algorithms defined as a finite system of procedures by applying the Banach contraction principle both in a suitable product of fuzzy quasi-metrics defined on the domain of words and in the product quasi-metric space of complexity spaces.

Keywords: Algorithm, Recurrence equation, Fuzzy quasi-metric, Domain of words, Complexity space, Banach contraction principle, Fixed point, Improver.

1 Introduction

Complexity analysis in algorithms theory classifies the cost of execution of computer programs. This is essential because such a classification allows us to decide on the feasibility of software solutions. For this means, asymptotic cost analysis is generally used (see [2] or [7]). This technique compares the order of magnitude of the cost of an algorithm with a known cost.

Denotational semantics theory has proved to be suitable for the complexity analysis of “Divide and Conquer” algorithms via the so-called complexity spaces as defined by Schellekens (see Section 6 of [24]). Furthermore, the existence (and uniqueness) of solution for the recurrence equations typically associated to Probabilistic Divide and Conquer algorithms and expoDC algorithms was deduced by using fixed point methods on complexity spaces in [9] and [21], respectively.

In the last years some authors have applied alternatives based on fixed point theorems in the domain of words, equipped with suitable bicomplete fuzzy quasi-metrics, to prove the existence and uniqueness of solution for the recurrence equations typically associated to Divide and Conquer algorithms and Quicksort algorithms ([19,22,23]).

* The authors acknowledge the support of the Spanish Ministry of Science and Innovation, under grant MTM2009-12872-C02-01.

Taking advantage of denotational semantics to define recursive algorithms, we here study the suitability of the Banach contraction principle in fuzzy quasi-metric spaces and in product complexity spaces for the complexity analysis of algorithms based on finite systems of recurrence equations (see Section 2 and Section 3, respectively).

Recall that, among all kinds of algorithms, recursive algorithms are those defined in terms of calls to the algorithm itself. In fact, recursiveness is an unifying theory for algorithmic problems based on recurrence equations.

Our study is motivated, in part, by the following algorithm, considered by Atkinson in [1] p. 16-17], which is defined as two procedures P and Q depending the one on the other such that:

```
function P(n)
  if n > 0 then
    Q(n-1); C; P(n-1); C; Q(n-1)

function Q(n)
  if n > 0 then
    P(n-1); C; Q(n-1); C; P(n-1); C; Q(n-1)
```

where C denotes any statements taking time independent of n .

In fact, the complexity analysis of that algorithm was discussed using both approaches in [4] and in [3]. These studies complemented previous results of existing Divide and Conquer algorithms to decide on the suitability of both approaches.

Here we extend our previous results to study a system of n procedures depending each one on the others and forming a finite system of recurrence equations.

Concrete examples of this class of algorithms could be extracted from language theory scenarios; such a system of equations may represent mutually dependent rules of a grammar.

Another scenario where many cases can be found is enterprise object-oriented design. Such systems are usually multi-tiered ones and inside each of these tiers, collaboration and reusability is highly promoted. In this case we are modeling objects that rely the ones on the others. For a tier designed for user interface modeling this system would define window or pages navigation. For a services tier it would model the dependencies among business components. For a data access tier, it could model the relationships among database entities, and so on.

2 Fuzzy Approach to the Algorithms Cost Analysis

In order to show the existence and uniqueness of solution for a finite system of recurrence equations as defined in equation (9) below, we will apply the Banach fixed point theorem in a suitable product of fuzzy quasi-metrics defined on the domain of words. This technique was already used in [19] to prove the existence and uniqueness of solution for the recurrence equations associated with Quicksort, and Divide and Conquer algorithms.

2.1 Background

In the following, the letters \mathbb{N} and ω will denote the set of all positive integer numbers and the set of all non-negative integer numbers respectively. The supremum of a finite number of real numbers will be denoted by \max or by \vee , and the infimum of a finite number of real numbers will be denoted by \min or by \wedge , indistinctly.

Next we recall several concepts and facts which are basic in our study.

A quasi-metric on a set X is a function $d : X \times X \rightarrow [0, \infty)$ such that for all $x, y, z \in X$: (i) $x = y \Leftrightarrow d(x, y) = d(y, x) = 0$; (ii) $d(x, z) \leq d(x, y) + d(y, z)$.

A quasi-metric space is a pair (X, d) such that X is a set and d is a quasi-metric on X .

A quasi-metric space (X, d) such that $d(x, z) \leq \max\{d(x, y), d(y, z)\}$, for all $x, y, z \in X$, is called a non-Archimedean quasi-metric space, and d is said to be a non-Archimedean quasi-metric on X .

Each quasi-metric d on X induces a topology τ_d on X which has as a base the family of open balls $\{B_d(x, r) : x \in X, r > 0\}$, where $B_d(x, \varepsilon) = \{y \in X : d(x, y) < \varepsilon\}$ for all $x \in X$ and $\varepsilon > 0$.

Given a quasi-metric d on X , then the function d^{-1} defined by $d^{-1}(x, y) = d(y, x)$, is also a quasi-metric on X , called the conjugate of d , and the function d^s defined by $d^s(x, y) = \max\{d(x, y), d^{-1}(x, y)\}$ is a metric on X .

A quasi-metric space (X, d) is said to be bicomplete if (X, d^s) is a complete metric space. In this case we say that d is a bicomplete quasi-metric on X .

According to [26], a binary operation $* : [0, 1] \times [0, 1] \rightarrow [0, 1]$ is a continuous t-norm if $*$ satisfies the following conditions: (i) $*$ is associative and commutative; (ii) $*$ is continuous; (iii) $a * 1 = a$ for every $a \in [0, 1]$; (iv) $a * b \leq c * d$ whenever $a \leq c$ and $b \leq d$, with $a, b, c, d \in [0, 1]$.

Definition 1. [5][12]. A fuzzy quasi-metric on a set X is a pair $(M, *)$ such that $*$ is a continuous t-norm and M is a fuzzy set in $X \times X \times [0, \infty)$ such that for all $x, y, z \in X$:

- (KM1) $M(x, y, 0) = 0$.
- (KM2) $x = y$ if and only if $M(x, y, t) = M(y, x, t) = 1$ for all $t > 0$.
- (KM3) $M(x, z, t + s) \geq M(x, y, t) * M(y, z, s)$ for all $t, s \geq 0$.
- (KM4) $M(x, y, _) : [0, \infty) \rightarrow [0, 1]$ is left continuous.

Definition 2. [14]. A fuzzy metric on a set X is a fuzzy quasi-metric $(M, *)$ on X such that for each $x, y \in X$:

- (KM5) $M(x, y, t) = M(y, x, t)$ for all $t > 0$.

Definition 3. [5][12]. A fuzzy (quasi-)metric space is a triple $(X, M, *)$ such that X is a set and $(M, *)$ is a fuzzy (quasi-)metric on X .

Each fuzzy (quasi-)metric $(M, *)$ on a set X induces a topology τ_M on X which has as a base the family of open balls $\{B_M(x, \varepsilon, t) : x \in X, 0 < \varepsilon < 1, t > 0\}$, where $B_M(x, \varepsilon, t) = \{y \in X : M(x, y, t) > 1 - \varepsilon\}$.

If $(M, *)$ is a fuzzy quasi-metric on a set X , it is obvious that $(M^{-1}, *)$ is also a fuzzy quasi-metric on X , where M^{-1} is the fuzzy set in $X \times X \times [0, \infty)$ defined by

$$M^{-1}(x, y, t) = M(y, x, t). \tag{1}$$

Moreover, if we denote by M^i the fuzzy set in $X \times X \times [0, \infty)$ given by

$$M^i(x, y, t) = \min\{M(x, y, t), M^{-1}(x, y, t)\}, \tag{2}$$

then $(M^i, *)$ is, clearly, a fuzzy metric on X .

A fuzzy (quasi-)metric space $(X, M, *)$ such that

$$M(x, z, t) \geq \min\{M(x, y, t), M(y, z, t)\}, \tag{3}$$

for all $x, y \in X$ and $t > 0$, is said to be a non-Archimedean fuzzy (quasi-)metric space.

In [11], M. Grabiec introduced the following notions in order to obtain a fuzzy version of the classical Banach fixed point theorem:

A sequence $(x_n)_n$ in a fuzzy metric space $(X, M, *)$ is Cauchy provided that $\lim_{n \rightarrow \infty} M(x_n, x_{n+p}, t) = 1$ for each $t > 0$ and $p \in \mathbb{N}$.

A fuzzy metric space $(X, M, *)$ is complete provided that every Cauchy sequence in X is convergent.

In the sequel, and according to [13] and [29], a Cauchy sequence in Grabiec’s sense will be called G-Cauchy and a complete fuzzy metric space in Grabiec’s sense will be called G-complete.

On the other hand, following [27], a B-contraction on a fuzzy metric space $(X, M, *)$ is a self-map f on X such that there is a constant $\alpha \in (0, 1)$ satisfying

$$M(f(x), f(y), \alpha t) \geq M(x, y, t) \tag{4}$$

for all $x, y \in X, t > 0$.

Thus, Grabiec’s fixed point theorem can be formulated as follows.

Theorem 1. [11]. *Let $(X, M, *)$ be a G-complete fuzzy metric space such that $\lim_{t \rightarrow \infty} M(x, y, t) = 1$ for all $x, y \in X$. Then every B-contraction on X has a unique fixed point.*

The following quasi-metric generalizations of the notions of B-contraction and G-completeness were introduced in [19].

Definition 4. *A B-contraction on a fuzzy quasi-metric space $(X, M, *)$ is a self-map f on X such that there is a constant $\alpha \in (0, 1)$ satisfying*

$$M(f(x), f(y), \alpha t) \geq M(x, y, t) \tag{5}$$

for all $x, y \in X, t > 0$. The number α is then called a contraction constant of f .

Definition 5. *A sequence $(x_n)_n$ in a fuzzy quasi-metric space $(X, M, *)$ is called G-Cauchy if it is a G-Cauchy sequence in the fuzzy metric space $(X, M^i, *)$.*

Definition 6. *A fuzzy quasi-metric space $(X, M, *)$ is called G-bicomplete if the fuzzy metric space $(X, M^i, *)$ is G-complete.*

Then, Grabiec’s theorem was generalized to fuzzy quasi-metric spaces in [19] as follows.

Theorem 2. [19]. *Let $(X, M, *)$ be a G-bicomplete fuzzy quasi-metric space such that $\lim_{t \rightarrow \infty} M(x, y, t) = 1$ for all $x, y \in X$. Then every B-contraction on X has a unique fixed point.*

Since G-(bi) completeness is a very strong kind of completeness (see [10][29]), George and Veeramani introduced the following notions:

A sequence $(x_n)_n$ in a fuzzy metric space $(X, N, *)$ is a Cauchy sequence [10] if for each $\epsilon \in (0, 1)$, $t > 0$ there exists $n_0 \in \mathbb{N}$ such that $M(x_n, x_m, t) > 1 - \epsilon$ for all $n, m \geq n_0$.

A fuzzy metric space is complete provided that every Cauchy sequence is convergent.

Definition 7. *A fuzzy quasi-metric space $(X, M, *)$ is called bicomplete if the fuzzy metric space $(X, M^i, *)$ is complete.*

Then we have the following nice and useful fact for our approach.

Theorem 3. [19]. *Each bicomplete non-Archimedean fuzzy quasi-metric space is G-bicomplete.*

Let us recall [12] that if (X, d) is a (quasi-)metric space, then the pair (M_d, \wedge) is a fuzzy (quasi-)metric on X where M_d is the fuzzy set in $X \times X \times [0, \infty)$ given by $M_d(x, y, 0) = 0$, and, for $t > 0$, by

$$M_d(x, y, t) = \frac{t}{t + d(x, y)}. \tag{6}$$

The triple (X, M_d, \wedge) is called the standard fuzzy (quasi-)metric space.

Furthermore, we have that $(M_d)^{-1} = M_{d^{-1}}$ and $(M_d)^i = M_{d^s}$. In addition, topology τ_d , induced by d coincides with the topology τ_{M_d} induced by the fuzzy (quasi-)metric (M_d, \wedge) .

2.2 The Banach Fixed Point Theorem on Fuzzy Quasi-metric Spaces Applied to Algorithms Cost Analysis

We start this subsection by constructing a suitable non-Archimedean quasi-metric on the domain of words, which will be crucial in our approach.

The domain of words Σ^∞ ([15][16][18][25][28], etc) consists of all finite and infinite sequences (“words”) over a nonempty set (“alphabet”) Σ , ordered by the so-called information order \sqsubseteq on Σ^∞ , i.e., $x \sqsubseteq y \Leftrightarrow x$ is a prefix of y , where we assume that the empty sequence ϕ is an element of Σ^∞ .

For each $x, y \in \Sigma^\infty$ denote by $x \sqcap y$ the longest common prefix of x and y , and for each $x \in \Sigma^\infty$ denote by $\ell(x)$ the length of x . Thus $\ell(x) \in [1, \infty]$ whenever $x \neq \phi$, and $\ell(\phi) = 0$.

Given a nonempty alphabet Σ , Smyth introduced in [28] a non-Archimedean quasi-metric d_\sqsubseteq on Σ^∞ given by

$$d_\sqsubseteq(x, y) = 0 \text{ if } x \sqsubseteq y, \text{ and } d_\sqsubseteq(x, y) = 2^{-\ell(x \sqcap y)} \text{ otherwise,}$$

(see also [15][17][19], etc).

This quasi-metric has the advantage that its specialization order coincides with the order \sqsubseteq , and thus the quasi-metric space $(\Sigma^\infty, d_{\sqsubseteq})$ preserves the information provided by \sqsubseteq . Moreover, the metric $(d_{\sqsubseteq})^s$ is given by

$$(d_{\sqsubseteq})^s(x, y) = 0 \text{ if } x = y, \text{ and } (d_{\sqsubseteq})^s(x, y) = 2^{-\ell(x \sqcap y)} \text{ otherwise,}$$

so that $(d_{\sqsubseteq})^s$ is exactly the celebrated Baire metric on Σ^∞ . Since the Baire metric is complete, it follows that $(\Sigma^\infty, d_{\sqsubseteq})$ is a bicomplete non-Archimedean quasi-metric space.

In order to apply techniques of fixed point for obtaining the existence and uniqueness of solution for the system of k recurrence equations associated to algorithms with k recurrence procedures, we shall combine the above results with some facts on the product of (non-Archimedean) fuzzy quasi-metric spaces that we present in the sequel.

Let us recall that given a quasi-metric d on a set X and a $k \in \mathbb{N}$, the product quasi-metric is the quasi-metric d^k on X^k defined by

$$d^k(x, y) = \max_{1 \leq j \leq k} d(x_j, y_j). \tag{7}$$

for all $x = (x_1, \dots, x_k), y = (y_1, \dots, y_k) \in X^k$.

In this case the quasi-metric space (X^k, d^k) is the (so-called) product quasi-metric space.

Similarly (compare [6]), given a fuzzy quasi-metric space $(M, *)$ on a set X and a $k \in \mathbb{N}$, the product fuzzy quasi-metric is the fuzzy quasi-metric $(M^k, *)$ on X^k defined by

$$M^k(x, y, t) = M(x_1, y_1, t) * \dots * M(x_k, y_k, t), \tag{8}$$

for all $x = (x_1, \dots, x_k), y = (y_1, \dots, y_k) \in X^k$, and $t \geq 0$.

In this case the fuzzy quasi-metric space $(X^k, M^k, *)$ is the (so-called) product fuzzy quasi-metric space.

Remark 1. Note that if (X, M_d, \wedge) is the standard fuzzy quasi-metric space of a quasi-metric space (X, d) , then $(X^k, (M_d)^k, \wedge)$ is the standard fuzzy quasi-metric space of the product quasi-metric space (X^k, d^k) .

Indeed, for $x = (x_1, \dots, x_k), y = (y_1, \dots, y_k) \in X^k$, and $t > 0$, we have

$$\begin{aligned} M_{d^k}(x, y, t) &= \frac{t}{t + d^k(x, y)} = \frac{t}{t + \max_{1 \leq j \leq k} d(x_j, y_j)} = \min_{1 \leq j \leq k} \frac{t}{t + d(x_j, y_j)} \\ &= \min_{1 \leq j \leq k} M_d(x_j, y_j, t) = (M_d)^k(x, y, t). \end{aligned}$$

Now let Σ be a non-empty alphabet, and consider the product fuzzy quasi-metric space $((\Sigma^\infty)^k, (d_{\sqsubseteq})^k)$.

Then, since $(\Sigma^\infty, d_{\sqsubseteq})$ is bicomplete and non-Archimedean, it follows that $((\Sigma^\infty)^k, (d_{\sqsubseteq})^k)$ is bicomplete and non-Archimedean, and consequently the standard fuzzy quasi-metric space $((\Sigma^\infty)^k, (M_{d_{\sqsubseteq}})^k, \wedge)$ of $((\Sigma^\infty)^k, (d_{\sqsubseteq})^k)$ is bicomplete and non-Archimedean.

Hence, the following result is a direct consequence of Theorems 2 and 3.

Theorem 4. *$((\Sigma^\infty)^k, (M_{d_\sqsubseteq})^k, \wedge)$ is a G-bicomplete fuzzy quasi-metric space such that $\lim_{t \rightarrow \infty} (M_{d_\sqsubseteq})^k(x, y, t) = 1$ for all $x, y \in (\Sigma^\infty)^k$. Therefore, every B-contraction on this space has a unique fixed point.*

In the sequel, the alphabet Σ will be the set of all non-negative real numbers, and for each $x \in \Sigma^\infty$ we will denote by $(x)_n$ its $(n + 1)$ -th letter. For instance, if $\ell(x) = 3$, $x := (x_0)(x_1)(x_2)$.

Let the finite system of k recurrence equations T_1, \dots, T_k , defined by

$$\begin{cases} T_1(n) = a_{11}T_1(n - 1) + a_{12}T_2(n - 1) + \dots + a_{1k}T_k(n - 1) + c_1 \\ T_2(n) = a_{21}T_1(n - 1) + a_{22}T_2(n - 1) + \dots + a_{2k}T_k(n - 1) + c_2 \\ \dots\dots\dots \\ T_k(n) = a_{k1}T_1(n - 1) + a_{k2}T_2(n - 1) + \dots + a_{kk}T_k(n - 1) + c_k \end{cases} \tag{9}$$

with $n \in \mathbb{N}$, $a_{ij} \geq 0$, and $c_i \geq 0$, for $i, j = 1, \dots, k$; and $T_i(0) > 0$, for $i = 1, \dots, k$.

Construct a functional $\Phi : (\Sigma^\infty)^k \rightarrow (\Sigma^\infty)^k$ as follows:

$$\Phi(x_1, \dots, x_k) = (u_1, \dots, u_k), \tag{10}$$

where, for $i = 1, \dots, k$, $\ell(u_i) = 1 + \min_{1 \leq j \leq k} \ell(x_j)$, and

$$(u_i)_0 = T_i(0), \text{ and}$$

$$(u_i)_n = a_{i1}(x_1)_{n-1} + a_{i2}(x_2)_{n-1} + \dots + a_{ik}(x_k)_{n-1} + c_i,$$

whenever $0 < n \leq 1 + \min_{1 \leq j \leq k} \ell(x_j)$.

We shall show that Φ is a B-contraction on $((\Sigma^\infty)^k, (M_{d_\sqsubseteq})^k, \wedge)$, with contraction constant $1/2$.

Let $(x_1, \dots, x_k), (y_1, \dots, y_k) \in (\Sigma^\infty)^k$, and $\Phi(x_1, \dots, x_k) = (u_1, \dots, u_k)$, $\Phi(y_1, \dots, y_k) = (v_1, \dots, v_k)$.

If $u_i \sqsubseteq v_i$ for all $i \in \{1, \dots, k\}$, then

$$\begin{aligned} M_{(d_\sqsubseteq)^k}(\Phi(x_1, \dots, x_k), \Phi(y_1, \dots, y_k), t/2) &= \min_{1 \leq i \leq k} M_{d_\sqsubseteq}(u_i, v_i, t/2) \\ &= \min_{1 \leq i \leq k} \frac{t/2}{t/2 + d_\sqsubseteq(u_i, v_i)} = 1, \end{aligned}$$

for all $t > 0$.

Otherwise, we have

$$\ell(u_i \sqcap v_i) \geq 1 + \min_{1 \leq j \leq k} \ell(x_j \sqcap y_j).$$

for all $i \in \{1, \dots, k\}$.

Therefore

$$\begin{aligned}
 M_{(d_{\sqsubseteq})^k}(\Phi(x_1, \dots, x_k), \Phi(y_1, \dots, y_k), t/2) &= \min_{1 \leq i \leq k} M_{d_{\sqsubseteq}}(u_i, v_i, t/2) \\
 &= \min_{1 \leq i \leq k} \frac{t/2}{t/2 + d_{\sqsubseteq}(u_i, v_i)} \\
 &= \min_{1 \leq i \leq k} \frac{t}{t + 2^{-\ell(u_i \sqcap v_i) + 1}} \\
 &\geq \frac{t}{t + 2^{-\min_{1 \leq i \leq k} \ell(x_i \sqcap y_i)}} \\
 &= \min_{1 \leq i \leq k} \frac{t}{t + 2^{-\ell(x_i \sqcap y_i)}} \\
 &= \min_{1 \leq i \leq k} M_{d_{\sqsubseteq}}(x_i, y_i, t) \\
 &= M_{(d_{\sqsubseteq})^k}((x_1, \dots, x_k), (y_1, \dots, y_k), t).
 \end{aligned}$$

So, by Remark 1,

$$(M_{d_{\sqsubseteq}})^k(\Phi(x_1, \dots, x_k), \Phi(y_1, \dots, y_k), t/2) \geq (M_{d_{\sqsubseteq}})^k(x_1, \dots, x_k), (y_1, \dots, y_k), t),$$

for all $t > 0$.

Hence, by Theorem 4 there exists a unique $z = (z_1, \dots, z_k) \in (\Sigma^\infty)^k$ such that $\Phi(z) = z$. Consequently z is the unique solution for the system 9.

Remark 2. As mentioned in Section 1 and following Atkinson [11 p. 16-17], the execution times $S(n)$ and $T(n)$ of $P(n)$ and $Q(n)$, satisfy, at least approximately, the recurrences

$$\begin{aligned}
 S(n) &= S(n - 1) + 2T(n - 1) + K_1, \\
 \text{and} & \tag{11}
 \end{aligned}$$

$$T(n) = 2S(n - 1) + 2T(n - 1) + K_2,$$

for $n \in \mathbb{N}$, and with K_1, K_2 , non-negative constants. Note that recurrences S and T are a particular case of the system 9 for $k = 2$.

Remark 3. In practice, one actually works on the set Σ^F of all finite words (over the alphabet $[0, \infty)$), that endowed with the restriction of $(M_{d_{\sqsubseteq}}, \wedge)$ provides a non-Archimedean fuzzy quasi-metric space which, obviously, is not bicomplete. In fact the product space $((\Sigma^F)^k, (M_{d_{\sqsubseteq}})^k, \wedge)$ is also a non-bicomplete non-Archimedean fuzzy quasi-metric space. However, for each $x \in (\Sigma^F)^k$, the sequence of iterations $(\Phi^m(x))_m$, is a Cauchy sequence in the complete fuzzy metric space $((\Sigma^\infty)^k,$

$\left((M_{d_{\square}})^k \right)^i, \wedge$) by the property of B-contraction of Φ stated above, and thus it converges to an element $y = (y_1, \dots, y_k)$ with $\ell(y_i) = \infty$ for $i = 1, \dots, k$, which is, in fact, the solution for the recurrence equations of system (9).

3 Complexity Spaces Approach to the Algorithms Cost Analysis

Schellekens introduced [24] the complexity (quasi-metric) space based on the Smyth completion [28] in order to construct a suitable mathematical model for the complexity analysis of algorithms. In fact, he proved in Section 6 of [24] the existence and uniqueness of solution for the recurrence equations associated to “Divide and Conquer” algorithms by applying a quasi-metric version of the Banach fixed point theorem to the complexity space. Recently it was shown in [9] that Schellekens’ technique can be successfully systematized to deduce the existence and uniqueness of solution for the recurrence equations associated to “Probabilistic Divide and Conquer” algorithms, and for the recurrence inequations associated to expoDC algorithms, respectively (see [8] and Section 7.7 of [2] for a study of such algorithms).

Here we show that the complexity space also provides an efficient framework to prove the existence and uniqueness of solution for the system of recurrence equations associated to a class of algorithms with a finite number of recurrence procedures as defined in (9). To this end, we will need to apply the Banach fixed point theorem to the “product complexity space” instead to the original one because this kind of algorithms involves several equations. Finally, we shall show that if (f_0, g_0) denotes the solution of the pair or recurrence equations S and T corresponding to the algorithm considered by Atkinson [1], then $f_0(n) \in \mathcal{O}(e^{2n})$ and $g_0(n) \in \mathcal{O}(e^{2n})$.

Let us recall that asymptotic notation introduces functions to denote the “order of” an algorithm f in the set of all possible functions. For a lower order threshold, \mathcal{O} -notation is used for any function $g : \omega \rightarrow [0, \infty)$:

$$\mathcal{O}(g(n)) = \{f : \omega \rightarrow [0, \infty) : \exists c > 0, n_0 \in \omega, \text{ such that } f(n) \leq cg(n) \text{ for all } n \geq n_0\}$$

3.1 Background

By a contraction map on a quasi-metric space (X, d) we mean a self-map f of X such that there is a constant $\alpha \in (0, 1)$ satisfying $d(fx, fy) \leq \alpha d(x, y)$ for all $x, y \in X$. The number α is then called a contraction constant of f .

It is clear that if f is a contraction map on a quasi-metric space (X, d) with contraction constant α then f is a contraction map on the metric space (X, d^s) with contraction constant α .

Therefore, the classical Banach contraction principle can be generalized to the quasi-metric setting as follows (see for instance [16, Lemma 2.4]).

Theorem 5. *Let f be a contraction map on a bicomplete quasi-metric space (X, d) . Then, for each $x \in X$, the sequence of iterations $(f^n x)_n$ is convergent in (X, d^s) to a point $x_0 \in X$ which is the unique fixed point of f .*

The so-called complexity space ([24]) is the quasi-metric space $(\mathcal{C}, d_{\mathcal{C}})$, where

$$\mathcal{C} = \left\{ f : \omega \rightarrow (0, \infty] : \sum_{n=0}^{\infty} 2^{-n} \frac{1}{f(n)} < \infty \right\}, \tag{12}$$

and $d_{\mathcal{C}}$ is the quasi-metric on \mathcal{C} given by

$$d_{\mathcal{C}}(f, g) = \sum_{n=0}^{\infty} 2^{-n} \left(\left(\frac{1}{f(n)} - \frac{1}{g(n)} \right) \vee 0 \right) \tag{13}$$

for all $f, g \in \mathcal{C}$. (We adopt the convention that $1/\infty = 0$.)

The elements of \mathcal{C} are called complexity functions.

The following useful result is a consequence of [20, Theorem 1, and Remark on p. 317].

Theorem 6. *The complexity space $(\mathcal{C}, d_{\mathcal{C}})$ is bicomplete.*

3.2 The Banach Fixed Point Theorem on Product Complexity Spaces Applied to Algorithms Cost Analysis

In order to simplify some formulas and equations which will be obtained in our next theorem, we re-write system (8) as

$$T_i(n) = \sum_{j=1}^k a_{ij} T_j(n-1) + c_i, \quad i = 1, \dots, k. \tag{14}$$

Moreover, we assume that $a_{ij} \neq 0$, for all $i, j \in \{1, \dots, k\}$, because, in this context, the rest of cases is a simplification of this one.

Theorem 7. *Let Ψ be the functional on \mathcal{C}^k defined, for each $f = (f_1, \dots, f_k) \in \mathcal{C}^k$, by*

$$\Psi(f)(0) = (T_1(0), \dots, T_k(0)) \quad \text{and} \tag{15}$$

$$\Psi(f)(n) = \left(\sum_{j=1}^k a_{1j} T_j(n-1) + c_1, \dots, \sum_{j=1}^k a_{kj} T_j(n-1) + c_k \right),$$

for all $n \in \mathbb{N}$.

If $\alpha < 2$, where $\alpha = \max_{1 \leq i \leq k} (\sum_{j=1}^k (1/a_{ij}))$, then:

(1) Ψ is a monotone increasing contraction on $(\mathcal{C}^k, (d_{\mathcal{C}})^k)$ with contraction constant $\alpha/2$.

(2) Ψ has a unique fixed point.

Proof

(1) Let $f = (f_1, \dots, f_k) \in \mathcal{C}^k$. Put $\Psi(f) = u = (u_1, \dots, u_k)$. Then, for each $i \in \{1, \dots, k\}$,

$$\begin{aligned} \sum_{n=1}^{\infty} 2^{-n} \frac{1}{u_i(n)} &= \sum_{n=1}^{\infty} 2^{-n} \frac{1}{a_{i1}f_1(n-1) + \dots + a_{ik}f_k(n-1) + c_i} \\ &\leq \frac{1}{a_{i1}} \sum_{n=0}^{\infty} 2^{-n} \frac{1}{f_1(n)} < \infty. \end{aligned}$$

Thus $u_i \in \mathcal{C}$ and consequently $\Psi(f) \in \mathcal{C}^k$.

Now let $f, g \in \mathcal{C}^k$ be such that $f \leq g$. Thus $f_i \leq g_i, i = 1, \dots, k$, where $f = (f_1, \dots, f_k)$ and $g = (g_1, \dots, g_k)$. It is straightforward to check that then $\Psi(f) \leq \Psi(g)$.

Next we show that for each $f = (f_1, \dots, f_k) \in \mathcal{C}^k$ and $g = (g_1, \dots, g_k) \in \mathcal{C}^k$, one has

$$(d_{\mathcal{C}})^k(\Psi(f), \Psi(g)) \leq \frac{\alpha}{2}(d_{\mathcal{C}})^k(f, g).$$

Indeed, for $f, g \in \mathcal{C}^k$, and $\Psi(f) = u = (u_1, \dots, u_k), \Psi(g) = v = (v_1, \dots, v_k)$, we have

$$(d_{\mathcal{C}})^k(\Psi(f), \Psi(g)) = \max_{1 \leq i \leq k} d_{\mathcal{C}}(u_i, v_i),$$

and, for each $i \in \{1, \dots, k\}$,

$$\begin{aligned} d_{\mathcal{C}}(u_i, v_i) &= \sum_{n=0}^{\infty} 2^{-n} \left(\left(\frac{1}{v_i(n)} - \frac{1}{u_i(n)} \right) \vee 0 \right) \\ &= \sum_{n=1}^{\infty} 2^{-n} \left(\left[\left(\sum_{j=1}^k a_{ij}g_j(n-1) + c_i \right)^{-1} - \left(\sum_{j=1}^k a_{ij}f_j(n-1) + c_i \right)^{-1} \right] \vee 0 \right) \\ &= \sum_{n=1}^{\infty} 2^{-n} \left(\left[\left(\sum_{j=1}^k a_{ij}(f_j(n-1) - g_j(n-1)) \right) \cdot \left(\sum_{j=1}^k a_{ij}^2 f_j(n-1)g_j(n-1) \right)^{-1} \right] \vee 0 \right) \\ &\leq \sum_{n=1}^{\infty} 2^{-n} \left(\sum_{j=1}^k \left(\frac{f_j(n-1) - g_j(n-1)}{a_{ij}f_1(n-1)g_j(n-1)} \vee 0 \right) \right) \\ &= \frac{1}{2} \sum_{n=0}^{\infty} 2^{-n} \left(\sum_{j=1}^k \frac{1}{a_{ij}} \left(\left(\frac{1}{g_j(n)} - \frac{1}{f_j(n)} \right) \vee 0 \right) \right) \\ &= \frac{1}{2} \sum_{j=1}^k \frac{1}{a_{i1}} d_{\mathcal{C}}(f_j, g_j) \\ &\leq \frac{1}{2} \left(\sum_{j=1}^k \frac{1}{a_{ij}} \right) \cdot \max_{1 \leq i \leq k} d_{\mathcal{C}}(f_i, g_i) \\ &\leq \frac{\alpha}{2} (d_{\mathcal{C}})^k(f, g). \end{aligned}$$

Consequently

$$(d_C)^k(\Psi(f), \Psi(g)) \leq \frac{\alpha}{2}(d_C)^k(f, g).$$

We conclude that Ψ is a contraction on $(\mathcal{C}^k, (d_C)^k)$ with contraction constant $\alpha/2$.

(2) Since (\mathcal{C}, d_C) is bicomplete (Theorem 6), it follows that $(\mathcal{C}^k, (d_C)^k)$ is also bicomplete. Then, by Theorem 5, Ψ has a unique fixed point which is obviously the solution of system (9). □

We finish the paper by showing that if (f_0, g_0) denotes the (unique) solution of the recurrence equations S and T associated with the algorithm discussed by Atkinson [1], then $f_0(n) \in \mathcal{O}(e^{2n})$ and $g_0(n) \in \mathcal{O}(e^{2n})$.

This will be done by constructing an appropriate element of \mathcal{C}^2 ($\mathcal{C} \times \mathcal{C}$ in the sequel) for which Φ is an improver.

To this end, the following extension to our context of Definition 6.2 of [24] will be needed.

Definition 8. *A functional Φ from $(\mathcal{C} \times \mathcal{C}, d_C \times d_C)$ into itself is an improver with respect to an element $(f, g) \in \mathcal{C} \times \mathcal{C}$ if for each $n \in \omega$, $\Phi^{n+1}(f, g) \leq \Phi^n(f, g)$.*

Note that if Φ is monotone increasing (i.e., $\Phi(f_1, g_1) \leq \Phi(f_2, g_2)$ whenever $f_1 \leq f_2$ and $g_1 \leq g_2$), to show that Φ is an improver with respect to (f, g) it suffices to verify that $\Phi(f, g) \leq (f, g)$.

Intuitively (compare, for instance, [9, p. 348]), an improver is a functional that corresponds to a transformation on algorithms and satisfies the following condition: the iterative applications of the transformation to a given algorithm yield an improved algorithm at each step of the iteration.

Put $c = (S(0) + T(0) + K_1 + K_2)(e^2 - 4)^{-1}$, and let $u, v : \omega \rightarrow (0, \infty)$ given by $u(0) = S(0)$, $v(0) = T(0)$, and $u(n) = v(n) = ce^{2n}$ for all $n \in \mathbb{N}$.

Clearly $u, v \in \mathcal{C}$. Next we show that $\Phi((u, v)) \leq (u, v)$, and thus Φ is an improver with respect to (u, v) .

Indeed, we have

$$\begin{aligned} \Phi((u, v))(0) &= (S(0), T(0)) = (u(0), v(0)), \\ \Phi((u, v))(1) &= (u(0) + 2v(0) + K_1, 2u(0) + 2v(0) + K_2) \\ &= (S(0) + 2T(0) + K_1, 2S(0) + 2T(0) + K_2) \\ &\leq ((S(0) + T(0) + K_1 + K_2) \frac{e^2}{e^2 - 4}, \\ &\quad (S(0) + T(0) + K_1 + K_2) \frac{e^2}{e^2 - 4}) \\ &= (ce^2, ce^2) \\ &= (u(1), v(1)) \\ &= (u, v)(1). \end{aligned}$$

and for $n > 1$,

$$\begin{aligned}
 \Phi((u, v))(n) &= (u(n-1) + 2v(n-1) + K_1, 2u(n-1) + 2v(n-1) + K_2) \\
 &= (ce^{2(n-1)} + 2ce^{2(n-1)} + K_1, 2ce^{2(n-1)} + 2ce^{2(n-1)} + K_2) \\
 &\leq (4ce^{2(n-1)} + K_1 + K_2, 4ce^{2(n-1)} + K_1 + K_2) \\
 &\leq (4ce^{2(n-1)} + c(e^2 - 4), 4ce^{2(n-1)} + c(e^2 - 4)) \\
 &\leq (ce^{2(n-1)}(4 + (e^2 - 4)), ce^{2n}(4 + (e^2 - 4))) \\
 &= (ce^{2n}, ce^{2n}) \\
 &= (u, v)(n).
 \end{aligned}$$

Since Φ is increasing it follows that $\Phi^{n+1}((u, v)) \leq \Phi^n((u, v))$ for all $n \in \omega$. Therefore, from the fact (see Theorem 5) that $(\Phi^n((u, v)))_n$ converges to (f_0, g_0) in $(\mathcal{C} \times \mathcal{C}, (d_{\mathcal{C}} \times d_{\mathcal{C}})^s)$, it follows that $(f_0, g_0) \leq (u, v)$. Consequently $f_0(n) \in \mathcal{O}(e^{2n})$ and $g_0(n) \in \mathcal{O}(e^{2n})$.

References

1. Atkinson, M.D.: The Complexity of Algorithms. In: Computing Tomorrow: Future Research Directions in Computer Science, pp. 1–20. Cambridge, Univ. Press, New York (1996)
2. Brassard, G., Bratley, P.: Fundamentals of Algorithms. Prentice Hall (1996)
3. Castro-Company, F., Romaguera, S., Tirado, P.: An Application of the Banach Contraction Principle on the Product of Complexity Spaces to the Study of Certain Algorithms with Two Recurrence Procedures. In: Vigo-Aguiar, J. (ed.) Proceedings of the 2010 International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE 2010), Almería, Spain, vol. 5, pp. 978–984 (2010) ISBN 978-84-613-5510-5
4. Castro-Company, F., Romaguera, S., Tirado, P.: Application of the Banach Fixed Point Theorem on Fuzzy Quasi-Metric Spaces to Study the Cost of Algorithms with Two Recurrence Equations. In: IJCCI 2010 2nd International Joint Conference on Computational Intelligence (2010) ISBN 978-989-8425-32-4
5. Cho, Y.J., Grabiec, M., Radu, V.: On non Symmetric Topological and Probabilistic Structures. Nova Sci. Publ. Inc., New York (2006)
6. Cho, Y.J., Grabiec, M., Taleshian, A.A.: Cartesian product of PQM-spaces. J. Nonlinear Sci. Appl. 2, 60–70 (2009)
7. Cormen, T.H., Leiserson, C.E., Stein, C., Rivest, R.L.: Introduction to Algorithms, 3rd edn. MIT Press (2001)
8. Flajolet, P.: Analytic Analysis of Algorithms. In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 186–210. Springer, Heidelberg (1992)
9. García-Raffi, L.M., Romaguera, S., Schellekens, M.: Applications of the Complexity Space to the General Probabilistic Divide and Conquer Algorithms. J. Math. Anal. Appl. 348, 346–355 (2008)
10. George, A., Veeramani, P.: On Some Results in Fuzzy Metric Spaces. Fuzzy Sets and Systems 64, 395–399 (1994)
11. Grabiec, M.: Fixed Points in Fuzzy Metric Spaces. Fuzzy Sets and Systems 27, 385–389 (1988)

12. Gregori, V., Romaguera, S.: Fuzzy Quasi-Metric Spaces. *Appl. Gen. Topology* 5, 129–136 (2004)
13. Gregori, V., Sapena, A.: On Fixed Point Theorems in Fuzzy Metric Spaces. *Fuzzy Sets and Systems* 125, 245–253 (2002)
14. Kramosil, I., Michalek, J.: Fuzzy Metrics and Statistical Metric Spaces. *Kybernetika* 11, 326–334 (1975)
15. Künzi, H.P.A.: Nonsymmetric Topology. In: *Proceedings of the Colloquium on Topology*. Szekszárd, Colloq. Math. Soc. János Bolyai Math. Studies, Hungary, vol 4, pp. 303–338 (1995)
16. Matthews, S.G.: Partial Metric Topology. In: *Proceedings of the 8th Summer Conference on General Topology and Applications*, *Ann. New York Acad. Sci.*, vol. 728, pp. 183–197 (1994)
17. Rodríguez-López, J., Romaguera, S., Valero, O.: Denotational Semantics for Programming Languages, Balanced Quasi-Metrics and Fixed Points. *Internat. J. Comput. Math.* 85, 623–630 (2008)
18. Romaguera, S., Schellekens, M.: Partial Metric Monoids and Semivaluation Spaces. *Topology Appl.* 153, 948–962 (2005)
19. Romaguera, S., Sapena, A., Tirado, P.: The Banach Fixed Point Theorem in Fuzzy Quasi-Metric Spaces with Application to the Domain of Words. *Topology Appl.* 154, 2196–2203 (2007)
20. Romaguera, S., Schellekens, M.: Quasi-Metric Properties of Complexity Spaces. *Topology Appl.* 98, 311–322 (1999)
21. Romaguera, S., Schellekens, M., Tirado, P., Valero, O.: Contraction Maps on Complexity Spaces and ExpoDC Algorithms. In: *Proceedings of the International Conference of Computational Methods in Sciences and Engineering ICCMSE 2007*, AIP Conference Proceedings, vol. 963, pp. 1343–1346 (2007)
22. Romaguera, S., Tirado, P.: Contraction Maps on IFQM-Spaces with Application to Recurrence Equations of Quicksort. *Electronic Notes in Theoret. Comput. Sci.* 225, 269–279 (2009)
23. Saadati, R., Vaezpour, S.M., Cho, Y.J.: Quicksort Algorithm: Application of a Fixed Point Theorem in Intuitionistic Fuzzy Quasi-Metric Spaces at a Domain of Words. *J. Comput. Appl. Math.* 228, 219–225 (2009)
24. Schellekens, M.: The Smyth Completion: a Common Foundation for Denotational Semantics and Complexity Analysis. *Electronic Notes Theoret. Comput. Sci.* 1, 535–556 (1995)
25. Schellekens, M.: The Correspondence between Partial Metrics and Semivaluations. *Theoret. Comput. Sci.* 315, 135–149 (2004)
26. Schweizer, B., Sklar, A.: *Probabilistic Metric Spaces*. North-Holland, Amsterdam (1983)
27. Sehgal, V.M., Bharucha-Reid, A.T.: Fixed Points of Contraction Mappings on PM-spaces. *Math. Systems Theory* 6, 97–100 (1972)
28. Smyth, M.B.: Quasi-uniformities: Reconciling Domains with Metric Spaces. In: Main, M.G., Mislove, M.W., Melton, A.C., Schmidt, D. (eds.) *MFPS 1987*. LNCS, vol. 298, pp. 236–253. Springer, Heidelberg (1988)
29. Vasuki, R., Veeramani, P.: Fixed Point Theorems and Cauchy Sequences in Fuzzy Metric Spaces. *Fuzzy Sets and Systems* 135, 415–417 (2003)

Part III
Neural Computation

SVM-Based Object Detection Using Self-quotient ε -Filter and Histograms of Oriented Gradients

Mitsuharu Matsumoto*

The Education and Research Center for Frontier Science,
The University of Electro-Communications,
1-5-1, Chofugaoka, Chofu-shi, Tokyo, 182-8585, Japan
mitsuharu.matsumoto@ieee.org
<http://www.mm-labo.org>

Abstract. This paper presents noise robust object detection using self-quotient ε -filter (SQEF) and histograms of oriented gradients (HOG). Although object detection combining HOG and support vector machine (SVM) is a promising approach, when the images are corrupted with noise, it is difficult to realize a good detection performance. To handle noise corrupted images, we employ SQEF, and apply it to object detection combining HOG and SVM. SQEF is an improved self-quotient filter (SQF), and can clearly extract features from the images not only when they have illumination variations but also when they are corrupted with noise. We confirmed the effectiveness of our approach by using human images and car images. Throughout the experiments, our approach can realize a robust object detection from noise corrupted images using the data trained by intact images without noise.

Keywords: Object detection, Histograms of oriented gradients, Self-quotient filter, Self-quotient ε -filter, Impulse noise, Feature extraction.

1 Introduction

Human and object detection in images is a challenging task and has a wide range of applications in industrial fields [1][2][3]. It is widely used in many applications such as image analysis, smart cars, and visual surveillance to behavioral analysis. The first requirement is a robust feature set that allows object form to be discriminated cleanly, even in backgrounds under different illumination. Histograms of oriented gradients (HOG) algorithm is a promising approach to match this requirement [4] and provides excellent performance compared to other existing feature sets including wavelets [5]. It is reminiscent of edge orientation [6], SIFT descriptors [7] and shape contexts [8]. Many studies have been reported concerning HOG. Although locally normalized HOG detectors are attractive approaches to detect the human from the image, it is difficult to detect them from the noise corrupted images because it uses local intensity gradients.

* This research was supported by the research grant of Support Center for Advanced Telecommunications Technology Research (SCAT), by the research grant of Foundation for the Fusion of Science and Technology, and by the Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Young Scientists (B), 20700168, 2008.

To solve the problem, we introduce self-quotient ε -filter (SQEF) [9], which is an advanced noise robust self-quotient filter (SQF) and propose a noise robust support vector machine (SVM)-based object detection combining SQEF and HOG. SQF is a simple nonlinear filter to extract the feature from an image [10]. It needs only an image, and can extract intrinsic lighting invariant property of an image, while removing extrinsic factor corresponding to the lighting. Feature extraction by SQF is simpler than that based on multi-scale smoothing [11]. SQF can extract the outline of the objects independent of shadow region. However, as it assumes that the image does not include noise, it can not extract the shape and texture when the noise damages the image. The noise influence becomes large due to the self-quotient effect of SQF.

SQEF is based on the idea of SQF and ε -filter [12][13]. ε -filter is a simple edge-preserving nonlinear filter. Although many studies have been reported to reduce the small amplitude noise while preserving the edge [14], it is considered that ε -filter is a promising approach due to its simple design. It does not need to have the signal and noise models in advance. It is easy to be designed and the calculation cost is small because it requires only switching and linear operation. We can clearly extract the feature from noise corrupted images by defining SQEF as the ratio of two different ε -filters. In this paper, we aim to reduce the noise influence by employing SQEF as preprocessing of HOG.

The rests of this paper are organized as follows. In section 2 we briefly introduce SQEF, and discuss the merits of SQEF compared to SQF. We also describe the algorithm of SVM-based object detection combining SQEF and HOG. Experimental results are shown to clarify the effectiveness of the proposed method for human detection from noise corrupted images compared to other approaches in section 3. We also show the results on car detection from noise corrupted images. A libsvm [15], MIT pedestrian test set [16], MIT CBCL car data [17] and standard image database (SIDBA) [18] are used as a SVM classifier, positive sample images for human detection, positive sample images for car detection and negative sample images for both experiments, respectively throughout the experiments. Conclusion is given in section 4.

2 Proposed Algorithm

This section gives the proposed algorithm. The procedure of our approaches is shown in Figure 1. In the proposed method, we first extract the feature from the noise corrupted image by using self-quotient ε -filter (SQEF) to eliminate not only illumination variations but also noise influence. Some examples are shown to clarify the difference between self-quotient filter (SQF) [10] and SQEF. Figure 2 shows the examples of filter output of SQEF to show its robust feature extraction from noise corrupted images. We also show the filter output of self-quotient filter (SQF). Fig. 2(a) shows a sample image from MIT pedestrian database [16]. Figs. 2(b) and 2(c) show the filter outputs of SQF and SQEF, respectively when we used the original image. On the other hand, Fig. 2(d) shows the sample image corrupted with 40% impulse noise. Figs. 2(e) and 2(f) show the filter outputs of SQF and SQEF, respectively when we used the impulse noise corrupted image. As shown in Fig. 2, both SQF and SQEF can extract the feature from the original image. However, SQF cannot extract its feature from the impulse noise corrupted image, while SQEF can extract the feature from the impulse noise corrupted image.

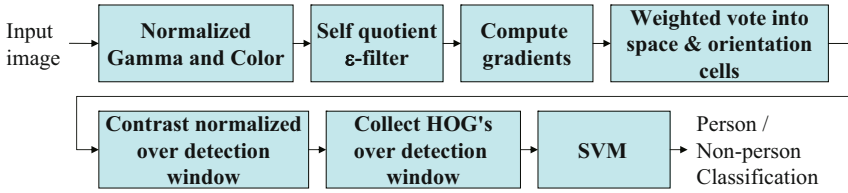


Fig. 1. An overview of our feature extraction and object detect chain

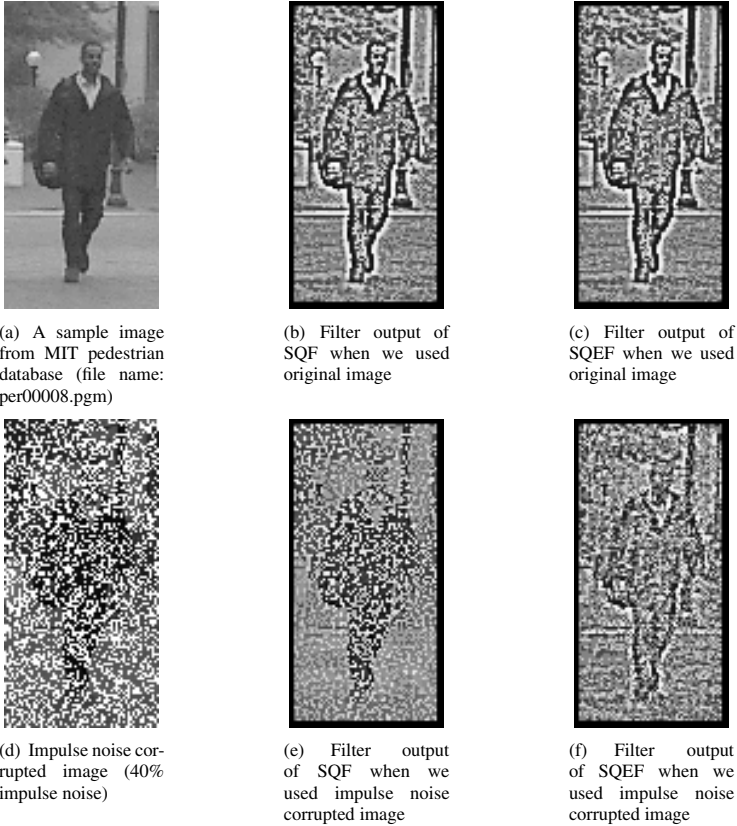


Fig. 2. Self-quotient image and self-quotient ε -filter from original image and impulse noise corrupted image

Let $x(i_1, i_2)$ be the image intensity at the point $\mathbf{i} = (i_1, i_2)$ in the image. The aim of SQF is to separate the intrinsic property and the extrinsic factor, and to remove the extrinsic factor. To handle the problem, SQF assumes that a smoothed version of an image has approximately the same illumination as the original one. In SQF, we first calculate the following equation:

$$y(i_1, i_2) = \frac{x(i_1, i_2)}{F[x(i_1, i_2)]}, \quad (1)$$

where $x(i_1, i_2)$ is the original image and F is the smoothing function. Due to the process of Eq. (1), the texture and edge can be extracted because the original image is divided by the smoothed image. However, SQF assumes that the image does not include the noise. When we consider the noise corrupted image, the noise is reduced in the smoothed images $F[x(i_1, i_2)]$, while the original image $x(i_1, i_2)$ includes the noise. As a result, the influence from the noise in SQF is emphasized very much as shown in Fig. 2 due to the self-quotient effect of SQF in Eq. (1).

A simple idea to solve the noise influence in SQF is to use two smoothed filters instead of original image as follows:

$$y(i_1, i_2) = \frac{F_1[x(i_1, i_2)]}{F_2[x(i_1, i_2)]}. \quad (2)$$

F_1 and F_2 should be different because the output always becomes 1 if F_1 and F_2 are the same smoothed filter.

However, even if we design SQF by using two different smoothed filters, not only the noise is smoothed but also the texture and shape are blurred. As the blur level of one smoothed filter is different from the other, it is also difficult to handle impulsive noise. Hence, we need to employ alternative filters, which can reduce the small amplitude noise effectively, while preserving the texture and shape information instead of simple smoothed filter. The alternative filters should be simple to keep the simplicity of SQF.

Based on the above prospects, self-quotient ε -filter (SQEF) is designed as follows:

$$y(i_1, i_2) = \frac{\Phi_{\varepsilon_1}[x(i_1, i_2)]}{\Phi_{\varepsilon_2}[x(i_1, i_2)]}, \quad (3)$$

where Φ_{ε} represents ε -filter described as follows:

$$\begin{aligned} z(i_1, i_2) = \\ \Phi_{\varepsilon}[x(i_1, i_2)] = x(i_1, i_2) + \\ \sum_{j_1=-K}^K \sum_{j_2=-K}^K a(j_1, j_2) F(x(i_1 + j_1, i_2 + j_2) - x(i_1, i_2)), \end{aligned} \quad (4)$$

where $a(j_1, j_2)$ represents the filter coefficient. $a(j_1, j_2)$ is usually constrained as follows:

$$\sum_{j_1=-K}^K \sum_{j_2=-K}^K a(j_1, j_2) = 1. \quad (5)$$

$F(x)$ is the nonlinear function described as follows:

$$|F(x)| \leq \varepsilon : -\infty \leq x \leq \infty, \quad (6)$$

where ε is a constant number constrained as follows.

$$0 \leq \varepsilon. \quad (7)$$

It should be noted that calculation cost of ε -filter is small because it requires only switching and linear operation. See the references [13] if the reader would like to know the details about ε -filter.

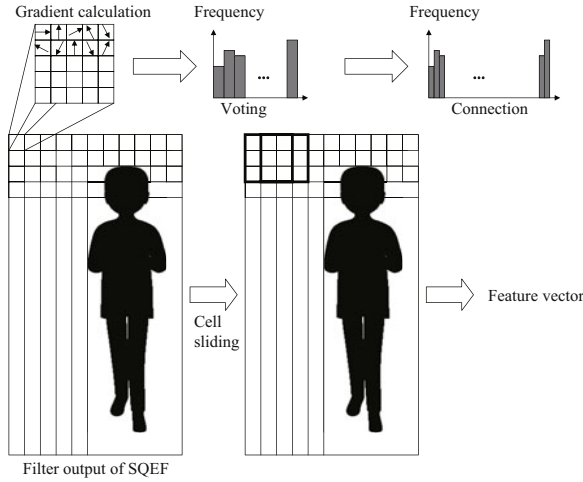


Fig. 3. Procedure of Histogram of Oriented Gradients (HOG) from SQEF output

When we apply SQEF to impulse noise corrupted image, it is considered that both ε -filters in SQEF keep the impulse noise in the image unlike when two smoothed filters are employed. Hence, when one filter output in SQEF is divided by the other filter in SQEF, the impulse noise effect is reduced by the self-quotient effects.

We next apply HOG procedure to SQEF outputs. Figure 3 shows the procedure of HOG from SQEF outputs. The method is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. As local object appearance and shape are kept in SQEF output, the gradient intensity and the gradient direction of SQEF are calculated for all the pixels as follows:

$$f_{i_1}(i_1, i_2) = y(i_1 + 1, i_2) - y(i_1 - 1, i_2) \quad (8)$$

$$f_{i_2}(i_1, i_2) = y(i_1, i_2 + 1) - y(i_1, i_2 - 1) \quad (9)$$

$$m(i_1, i_2) = \sqrt{f_{i_1}^2 + f_{i_2}^2} \quad (10)$$

$$\theta(i_1, i_2) = \arctan \frac{f_{i_2}}{f_{i_1}} \quad (11)$$

The basic idea of HOG is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions [4]. In practice, this is implemented by dividing the filter output into small spatial regions (“cells”), for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of cell. The obtained direction θ ($0^\circ \leq \theta \leq 180^\circ$) is divided with 20° intervals. 9 dimensional feature vector is generated by adding the gradient intensity $m(i_1, i_2)$. We then regard 3×3 cells as “Block” and generate many blocks by sliding on a pixel to pixel basis. The feature vector is finally obtained by combining all the feature vector. The obtained feature vector is adopted to SVM.

3 Experiments

3.1 Experiments on Human Detection

We first conducted the experiments on human detection using impulse noise corrupted images to show the effectiveness of the proposed method.

MIT pedestrian database and SIDBA were employed as image database. MIT pedestrian database contains 900 images. The size is 64 pixel \times 128 pixel. Some non person images were selected from standard image database (SIDBA). 900 64 pixel \times 128 pixel images were cut from them. We also prepared impulsive noise corrupted images by adding the impulse noise to the above 1800 images. Noise percentage changed from 10% to 40% with 10% intervals. Figure 4 shows original person / non-person images and its noise-corrupted version. Our aim is to detect human from these types of noise corrupted images not by using the data trained by the impulse noise corrupted image but by using the data trained by intact images without noise. As a SVM tool, we used libsvm, a library for support vector machines [15], and employed default setting and parameters throughout the experiments for simplicity.

In the experiments, we used original 450 pedestrian images from MIT pedestrian database and 450 non-person images from SIDBA. We tried to classify the impulse noise corrupted image by using the training data. The test images are the remaining 450 pedestrian images from MIT pedestrian database and the remaining 450 non-person images from SIDBA with impulse noise, which are different from the training images. For comparison, we also tested to classify them using the method combining HOG and SVM, and the method combining SQF, HOG and SVM. Figure 5 shows the recognition results. As shown in Fig 5, it was difficult to classify the images using the method combining HOG and SVM when the image was corrupted with the impulse noise. The results were still bad even when we used the method combining SQF, HOG and SVM. On the other hand, the proposed approach could detect human from noise corrupted images almost 100% using training data with intact images without noise.

3.2 Experiments on Car Detection

We second conducted the experiments on car detection using impulse noise corrupted images. We used 500 images from MIT CBCL Car Data [17] as car image database. The size is 128 pixel \times 128 pixel. Some non car images were selected from standard image database (SIDBA). 500 128 pixel \times 128 pixel images were cut from them. We also prepared impulsive noise corrupted images by adding the impulse noise to the above 1000 images the same as the previous experiment. Noise percentage changed from 10% to 40% with 10% intervals. Figure 6 shows original car / non-car images and its noise-corrupted version. Our aim is to detect car from these types of noise corrupted images not by using the data trained by the impulse noise corrupted image but by using the data trained by intact images without noise. We used the same SVM tool and the same parameter as the previous experiments.

In the experiments, we used original 250 car images from MIT CBCL Car Data and 250 non-car images from SIDBA. We tried to classify the impulse noise corrupted image by using the training data. The test images are the remaining 250 car images



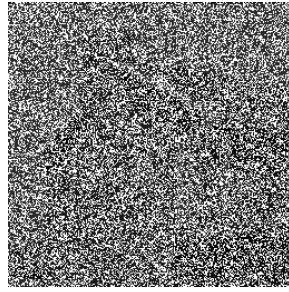
(a) Person image from MIT pedestrian database (per00003.pgm)



(b) Non-person image from SIDBA (Airplane)



(c) Person image from MIT pedestrian database with 40% impulse image (per00003.pgm)



(d) non-person image from SIDBA with 40% impulse noise (Airplane)

Fig. 4. Sample images of person image and non-person image (Original and noise corrupted images)

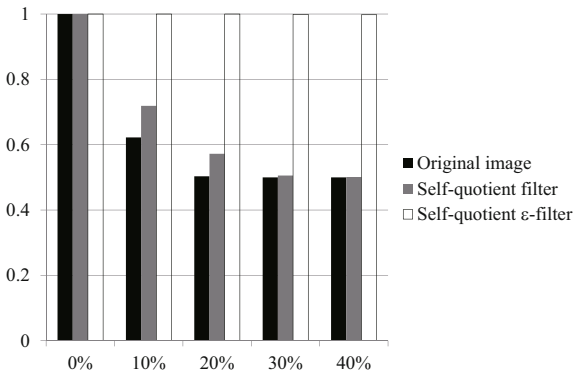


Fig. 5. Experimental results of human detection from impulse noise corrupted image

from MIT CBCL Car Data and the remaining 250 non-car images from SIDBA with impulse noise, which are different from the training images. For comparison, we also tested to classify them using the method combining HOG and SVM, and the method

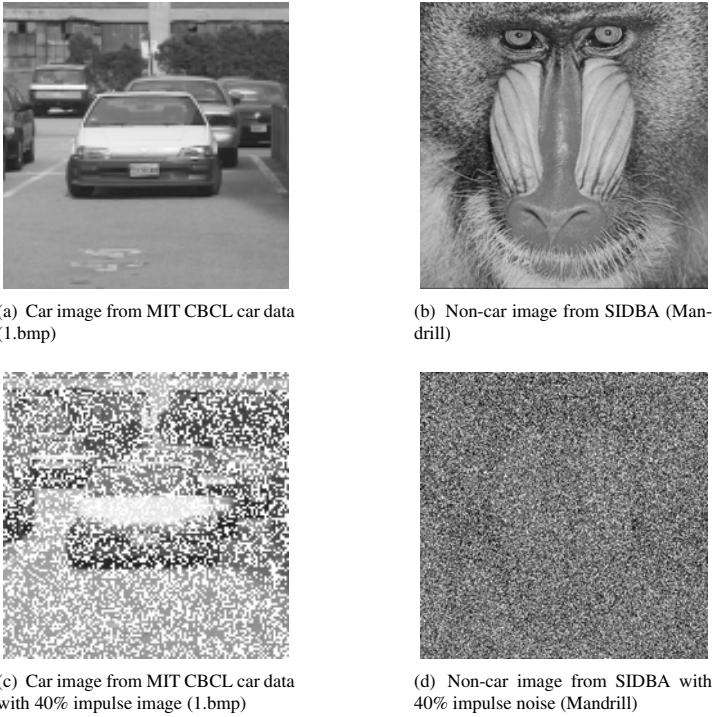


Fig. 6. Sample images of car image and non-car image (Original and noise corrupted images)

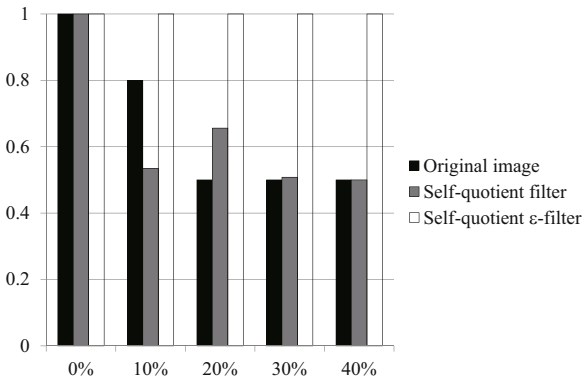


Fig. 7. Experimental results of car detection from impulse noise corrupted image

combining SQF, HOG and SVM the same as the previous experiment. Figure 7 shows the recognition results. As shown in Fig 7 it was difficult to classify the images using the method combining HOG and SVM when the image was corrupted with the impulse noise. The results were still bad even when we used the method combining SQF, HOG and SVM. On the other hand, the proposed approach could detect human from noise corrupted images about 100% using training data with intact images without noise.

4 Conclusions

This paper proposed a noise robust SVM-based object detection combining self-quotient ε -filter and histogram of oriented gradients. We compared the results of our approach to the results of HOG and SVM, and the results of SQF, HOG and SVM. Throughout the experiments, the proposed method could robustly detect pedestrians and cars from noise corrupted images using the training data with the clean image without noise, while it is difficult to detect objects using other approaches. Future works include the applications of our method to robot vision. Detailed study of effects of the parameters should also be required. We also would like to apply the proposed method to medical images to detect disease site from noise corrupted images.

Acknowledgements. This research was supported by the research grant of Support Center for Advanced Telecommunications Technology Research (SCAT), by the research grant of Foundation for the Fusion of Science and Technology, by Special Coordination Funds for Promoting Science and Technology, and by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B), 20700168, 2008.

References

1. Vadakkepat, P., Lim, P., Silva, L.C.D., Jing, L., Ling, L.L.: Multimodal Approach to Human-Face Detection and Tracking. *IEEE Trans. on Industrial Electronics* 55(3), 1385–1393 (2008)
2. Hsiao, P.-Y., Lu, C.-L., Fu, L.-C.: Multilayered Image Processing for Multiscale Harris Corner Detection in Digital Realization. *IEEE Trans. on Industrial Electronics* 57(5), 1799–1805 (2010)
3. Huang, W.-C., Wu, C.-H.: Adaptive color image processing and recognition for varying backgrounds and illumination conditions. *IEEE Trans. on Industrial Electronics* 45(2), 351–357 (1998)
4. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: *Proc. of Int'l Conf. on Computer Vision and Pattern Recognition*, pp. 886–893 (2005)
5. Mohan, A., Papageorgiou, C., Poggio, T.: Example-based object detection in images by components. *IEEE Trans. on Pattern Recognition and Machine Intelligence* 23, 349–361 (2001)
6. Freeman, W.T., Tanaka, K., Ohta, J., Kyuma, K.: Computer vision for computer games. In: *Proc. of Int'l Conf. on Automatic Face and Gesture Recognition*, pp. 100–105 (1996)
7. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int'l Journal of Computer Vision* 60, 91–110 (2004)
8. Belongie, S., Malik, J., Puzicha, J.: Matching shapes. In: *Proc. of Int'l Conf. on Computer Vision*, pp. 454–461 (2001)
9. Matsumoto, M.: Self-quotient -filter for feature extraction from noise corrupted image. *IEICE Transactions on Information and Systems* E93-D(11), 3066–3075 (2010)
10. Wang, H., Zhang, J.J., Li, S.Z., Wang, Y.: Shape and texture preserved non-photorealistic rendering. *Computer Animation and Virtual Worlds* (2004)
11. Gooch, B., Reinhard, E., Gooch, A.: Human facial illustrations: Creations and psychological evaluation. *ACM Transactions on Graphics* 23(1), 27–44 (2004)
12. Arakawa, K., Matsuura, K., Watabe, H., Arakawa, Y.: A method of noise reduction for speech signals using component separating ε -filters. *IEICE Trans. on Fundamentals* J85-A(10), 1059–1069 (2002)

13. Arakawa, K., Okada, T.: ϵ -separating nonlinear filter bank and its application to face image beautification. *IEICE Trans. on Fundamentals* J90-A(4), 52–62 (2005)
14. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: *Proc. IEEE Int'l Conf. on Computer Vision*, pp. 839–846 (1998)
15. Chang, C.-C., Lin, C.-J.: *LIBSVM: a library for support vector machines* (2001)
16. Oren, M., Papageorgiou, C.P., Sinha, P., Osuna, E., Poggio, T.: Pedestrian Detection Using Wavelet Templates. In: *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 193–199 (1997)
17. Papageorgiou, C., Poggio, T.: A Trainable System for Object Detection. *International J. of Computer Vision* 38(1), 15–33 (2000)
18. <http://sipi.usc.edu/database/>

Adaptive Control of Robot Systems with Simple Rules Using Chaotic Dynamics in Quasi-layered Recurrent Neural Networks

Ryosuke Yoshinaka, Masato Kawashima, Yuta Takamura, Hitoshi Yamaguchi, Naoya Miyahara, Kei-ichiro Nabeta, Yongtao Li, and Shigetoshi Nara

Electrical & Electronic Department,
Graduate School of Nat. Sci. & Tech., Okayama University,
3-1-1, Tsushima-naka, Okayama, 700-8530 Japan
nara@ele.okayama-u.ac.jp

Abstract. A novel idea of adaptive control with simple rules using chaotic dynamics in a recurrent neural network model and two kinds of quasi-layered recurrent neural network model have been proposed. Since chaos in brain was discovered in the context of brain function, the authors have claimed that chaos has complex functional potentialities and have presented the results of computer experiments which use chaos to solve several kinds of "ill-posed problems". The key idea is to harness the onset of complex nonlinear dynamics in dynamical systems. More specifically, attractor dynamics and chaotic dynamics in a recurrent neural network model are introduced by changing a system parameter, "connectivity" in one type of model and via "sensitive response of chaos to external inputs" in other models. In this report, we will show the following. (1) A global outline of our idea and our recurrent neural network models with neuro-chaos, (2) Several computer experiments on the use of the neuro-chaos recurrent neural network models for solving of 2-dimensional mazes by an autonomous robot, in the context of an ill-posed problem setting, (3) Hardware implementations of the computer experiments using robots with two-wheels or two-legs driven by a neuro chaos simulator. Successful results of maze-solving are shown not only for computer experiments but also for practical experiments, (4) A proposal for a pseudo-neuron device using semiconductor and opto-electronic technologies. The device is called a "dynamic self-electro optical effect devices (DSEED)", and it has the potential to be a "neuromorphic device" or even a "brainmorphic device". (5) A proto-type model of intra-brain communications between far distant neurons in the brain is proposed, from a heuristic point of view based on observations of neuron synchronization phenomena associated with advanced brain functioning.

Keywords: Complex dynamics, Chaos, Adaptive control, Quasi-layered recurrent neural network, Hardware implementation, Autonomous robot, Neuromorphic device, Brainmorphic device.

1 Introduction

Observations of chaotic dynamics in biological systems, in particular brains, suggest to us that there may be important relations between chaotic dynamics and the functions

of biological systems, for example information processing, regulation or control, intra-brain communications between far areas. [1] [12] [13] [14] [19].

The rapid progress of robotics has brought various robots into our life and industry. However, it is still difficult for robots to perform tasks adaptively in various environments as biological systems do. Conventional methodologies for robotics such as designing systems from many parts and elements, often fall into difficulties in the face of enormous complexity arising in the dynamics of systems with many degrees of freedom.

For these reasons, it has been suggested that information processing and control in biological systems could work with novel *dynamical mechanisms*. Many dynamical models have been proposed for biological mechanisms and analyzed by means of large-scale simulation or heuristic methods. Our work is based on a novel idea to harness the onset of complex nonlinear dynamics in information processing or control systems, and on studies of chaotic dynamics in neural networks from the functional viewpoint. First, Nara & Davis introduced chaotic dynamics into a recurrent neural network model (RNNM) by adjusting only one system parameter, the connectivity among neurons, and proposed that constrained chaos could be useful to solve complex problems such as ill-posed problems [4]. In functional experiments, chaotic dynamics was applied to solving a memory search task or an image synthesis task set in an ill-posed context [5] [6] [8]. Furthermore, the idea has been extended to challenging applications of chaotic dynamics to control. Chaotic dynamics in a recurrent neural network model was applied to a control task in which an object solves a two-dimensional maze to catch a target [10], or to capture a target moving along different trajectories [9]. From the results of computer experiments, we consider that chaotic dynamics could be useful not only in solving ill-posed problems but also in controlling systems with many degrees of freedom.

In the present paper, we develop this idea further and propose two types of quasi-layered RNNMs. The first type has an upper layer consisting of sensing neurons and a lower layer consisting of driving neurons, with chaotic dynamics used in both layers. The second type consists of an upper layer (sensory-neurons), an intermediate layer (inter-neurons) and a lower layer (motor-neurons). This approach is based on the work of Mikami and Nara who found that chaos has the property of sensitive response to external input [3]. The RNNM is applied to solving 2-dimensional mazes. We can find a corresponding example in biological behavior, the auditory behavior of a cricket, which is a typical ill-posed problem in a biological system [2]. Further developments are shown about the following topics. They are

- (a) application to a roving robot with two legs
- (b) application to an arm robot
- (c) proposal of a hardware device for a pseudo-neuron and a network of pseudo-neurons, and evaluation by computer experiments
- (d) making of actual hardware device using semiconductor and opto-electronic technologies
- (e) proposal of a proto-type model of intra-brain communications between far areas in a brain, which are observed as synchronization of firing patterns associated with advanced functioning.

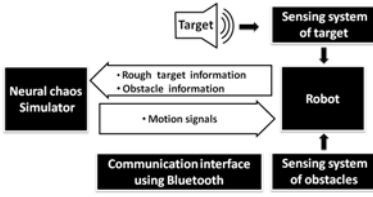


Fig. 1. The block diagram of our robot control system



Fig. 2. The roving robot in our hardware experiments

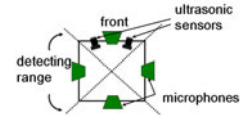


Fig. 3. The sensor configurations

2 Main Algorithm of Control System Using Chaos

Construction of Control System. The experimental system mainly consists of a roving robot with a micro processor unit (MPU), sensing system for sensing sound signals from the target and for detecting obstacles, a neural chaos simulator, a Bluetooth interface between the robot and the neural chaos simulator, and a target emitting a specific sound signal, which is like the song of a male cricket, as shown in Fig 1.

The robot with sensors is shown in Fig 2, 3 & 4. It has two driving wheels and one castor (2DW1C). The robot has six sensors which can be divided into two sub-systems. One is the sensing system for detecting obstacles, which consists of two ultrasonic sensors that give the robot the ability to detect whether an obstacle exists in front of the robot without actually touching it. The other is the sensing system for sensing sound signals from a target, which consists of four sets of directional microphone circuits that function as the *ears* of the robot. Four microphones are set facing four perpendicular directions, the front, the back, the left and the right of the robot, as shown in Fig 3. In our study, a loud speaker is employed to emit the sound from the target, a 3.6KHz signal sounding like a singing cricket. The sound signal from the target is picked up by the four microphone *ears*. These four sound signals are amplified, rectified, digitized, and transferred to the MPU. At the preliminary stage, these signals are compared to determine which direction has the strongest signal. In the future, we plan to input them to the upper layer (sensing neurons) in the quasi-layered RNNM. In the present study, We emphasize the two points. One is that the system for sensing the sound signal from a target does not give accurate information about the direction of the target, but only gives rough information about the direction of the target, with some uncertainty. The other is that, the signals input from the four microphones are not processed with complex techniques or methods. These are quite important differences between our work and other conventional robotic systems.

2.1 Context Setting of Solving Mazes

In the present study, the context for solving mazes is as follows.

- 1) Set obstacles unknown by the robot
- 2) Set a target emitting a sound signal

- 3) Acquire information for reaching the target • Check whether there is an obstacle which prevents the robot from moving forward, using ultrasonic sensors for obstacle detection • Obtain the rough direction of the target by comparing the signals from four microphones
- 4) Calculate movement increments at every time step of the neural network activity

3 Neural Chaos Simulator

A neural chaos simulator is used to simulate the dynamical behavior of a neural network, and works as the "brain" of the robot. Chaotic dynamics in the neural network cause the robot to generate complex motions which enable it to avoid obstacles in unknown environments such as mazes. In our study, we start from a simple RNNM, and develop it to a quasi-layered RNNM (abbreviated as QL-RNNM hereafter) in a heuristic approach to modeling the mechanism of a brain. In regard to QL-RNNM, we consider two kinds of QL-RNNM. The first type of QL-RNNM consists of sensory-neurons and motor-neurons, and the second type of QL-RNNM consists of sensory-, inter-, and motor-neurons.

3.1 Recurrent Neural Network Model

Our system uses a fully interconnected RNNM consisting of N binary neurons, with an updating rule defined by

$$S_i(t+1) = \text{sgn}\left(\sum_{j \in G_i(r)} W_{ij} S_j(t)\right) \quad (1)$$

$$\text{sgn}(u) = \begin{cases} +1 & u \geq 0; \\ -1 & u < 0. \end{cases}$$

- $S_i(t) = \pm 1 (i = 1 \sim N)$: the firing state of a neuron specified by index i at time t .
- W_{ij} : connection weight from the neuron S_j to the neuron S_i , W_{ii} is taken to be 0.
- $r (0 < r < N)$: fan-in number for neuron S_i , named connectivity
- $G_i(r)$: spatial configuration set of connectivity r for neuron S_i

At a certain time t , the state of neurons in the network can be represented as a N -dimensional state vector $\mathbf{S}(t)$, called a *state pattern*. The updating rule shows that time development of state pattern $\mathbf{S}(t)$ depends on the connection weight matrix W_{ij} and connectivity r . When there is full connectivity $r = N - 1$, appropriately determining W_{ij} by means of a kind of orthogonalized learning method [6] can be used to embed groups of N dimensional state patterns as cyclic memory attractors in N dimensional state space. Cyclic memory attractors consist of K patterns per cycle and we denote the number of cycles as L . For example, in Fig 4 we take $K = 6$, $L = 4$, and $N = 400$. In this case, the firing state of each of the $N = 20 \times 20 = 400$ neurons is represented by a black pixel or a white pixel. After the network evolves according to the updating rule for enough time steps, a randomly initial state pattern will converge into one of the embedded cyclic attractors.

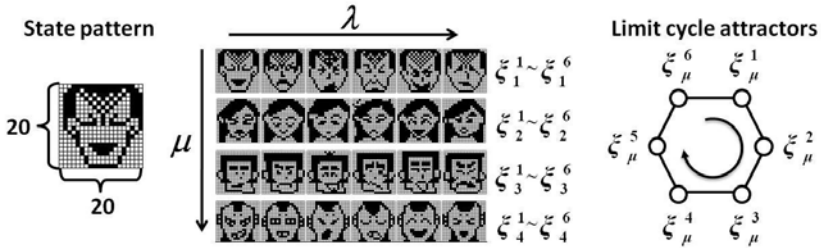


Fig. 4. An example of sets of patterns embedded as cyclic attractors : when connectivity $r = N - 1$, if $S(t)$ is ξ_1^1 , then the output sequence for $t > 1$ is $\xi_1^2, \xi_1^3, \dots, \xi_1^6, \xi_1^1, \dots$

Now if we reduce connectivity r by blocking signal transfer from other neurons, the attractors gradually become unstable, and the state dynamics changes from cyclic attractor dynamics to chaotic dynamics. In order to analyze the destabilizing process, we have calculated a bifurcation diagram for the overlap, (Fig 5), where the overlap is a one-dimensional projection of the state pattern $S(t)$ onto a certain reference pattern. The overlap $m(t)$ is defined as follows.

$$m(t) = \frac{1}{N} S(0) \cdot S(t) \tag{2}$$

$$t = Kp + t_0 \quad (p = 1, 2, \dots) \tag{3}$$

where $S(0)$ is the initial pattern which is taken as the reference pattern, and $S(t)$ is the state pattern at time step t . Because $m(t)$ is a normalized inner product, $-1 \leq m(t) \leq 1$. $m(t) = 1$ means that the present state pattern and the reference pattern are the same at every K steps. Figs. 5 and 6 show the calculated overlap $m(t)$ after $S(t)$ has evolved for a long time, for the upper layer and the lower layer, respectively.

3.2 Quasi-layered Recurrent Neural Network Model

A quasi-layered RNNM consists of an upper layer with N neurons and a lower layer with N neurons. The upper layer is updated by only self-recurrence. On the other hand, the lower layer is updated not only by self-recurrence but also by recurrent outputs of the upper layer. State pattern $S(t) = [x(t), y(t)]$, where $x(t) = \{x_i(t) = \pm 1 \mid i = 1, 2, \dots, N\}$ and $y(t) = \{y_i(t) = \pm 1 \mid i = 1, 2, \dots, N\}$. The updating rules of the two layers are defined by

$$x_i(t + 1) = \text{sgn} \left(\sum_{j \in G_u(r_u)} W_{ij}^{u-u} x_j(t) \right) \tag{4}$$

$$y_i(t + 1) = \text{sgn} \left(\sum_{j \in G_l(r_l)} [W_{ij}^{l-l} y_j(t) + W_{ij}^{u-l} x_j(t)] \right), \tag{5}$$

where u means upper layer and l means lower layer. W_{ij}^{u-u} is the connection weight from neuron x_j of the upper layer to neuron x_i of the upper layer, and W_{ij}^{u-l} and W_{ij}^{l-l} are defined similarly.

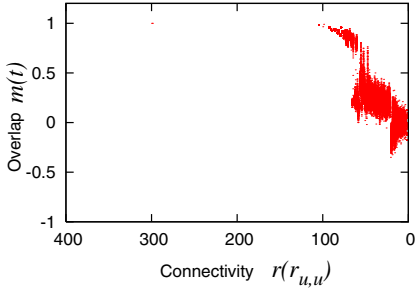


Fig. 5. The long-time behavior of the K-step overlap $m(t)$ in the upper layer. The horizontal axis is $r(r_{u,u})$ (0-399).

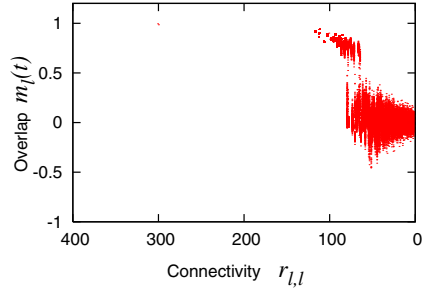


Fig. 6. The long-time behavior of the K-step overlap $m(t)$ in the lower layer. The horizontal axis is $r(r_{l,l})$ (0-399).

In the quasi-layered RNNM, if we take sufficiently large connectivity r ($r_{u,u} \simeq N, r_{u,l} \simeq N, r_{l,l} \simeq N$), by appropriately determining connection weight W_{ij} , a group of arbitrarily designed state patterns can be embedded as cyclic memory attractors. Certain cyclic memory attractors are embedded in each layer. Since three connectivity's r_{uu}, r_{ul}, r_{ll} affect the development of state pattern, for the upper layer and the lower layer, we have calculated the overlap $m(t)$ of a state pattern $\mathcal{S}(t)$ after it evolves for a long time. The overlap $m_u(t)$ of the upper layer as a function of connectivity $r_{u,u}$ is the same as in Fig 5.

Next, let us show examples of long time behaviors of $m_l(t)$ for a few cases of connectivity's, r_{uu}, r_{ul}, r_{ll} . They are shown in Fig 6 ($r_{l,l}$ dependence when $r_{u,l} = 0$), Fig 7 ($r_{l,l} + r_{u,l}$ dependence when $r_{u,l} \neq 0$). In regard to the latter case, let us show an example of time development of $m_u(t)$ and $m_l(t)$ in Fig 8 when $r_{u,u} = 40, r_{u,l} = 400, r_{l,l} = 399$, where the lower layer sensitively responds to the upper layer, depending on whether trajectories of the upper layer pass through states near the embedded attractors or far from them.

3.3 Quasi-layered Recurrent Neural Network Model Consisting of Sensory-, Inter-, and Motor-Neurons

The other quasi-layered RNNM consisting of three layers having N_s, N_i , and N_m neurons, respectively, is realized by adding new variables corresponding to inter-neurons. The upper layer is updated by only self-recurrence. On the other hand, the intermediate layer is updated not only by self-recurrence but also by recurrent outputs of the upper layer. The lower layer is also updated in the same way. State patterns are represented as $\mathcal{S}(t) = [\mathbf{x}(t), \mathbf{y}(t), \mathbf{z}(t)]$, where $\mathbf{x}(t) = \{x_i(t) = \pm 1 \mid i = 1, 2, \dots, N_s\}$, $\mathbf{y}(t) = \{y_i(t) = \pm 1 \mid i = 1, 2, \dots, N_i\}$ and $\mathbf{z}(t) = \{z_i(t) = \pm 1 \mid i = 1, 2, \dots, N_m\}$.

The updating rules of the three layers are defined by

$$x_i(t + 1) = \text{sgn} \left(\sum_{j \in G_u(r_u)} W_{ij}^{u-u} x_j(t) \right) \tag{6}$$

$$y_i(t + 1) = \text{sgn} \left(\sum_{j \in G_i(r_i)} [W_{ij}^{i-i} y_j(t) + W_{ij}^{u-i} x_j(t)] \right), \tag{7}$$

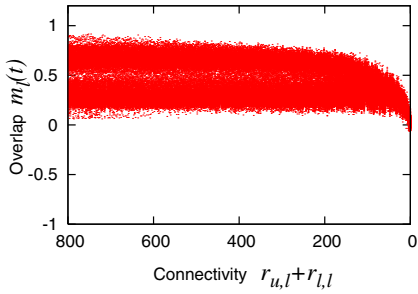


Fig. 7. The long-time behaviors of K -step overlaps $m_l(t)$ ($r_{u,u} = 40$, $r_{u,l} = 400$). The horizontal axis represents the reduced input connectivity to the lower layer, 0-799.

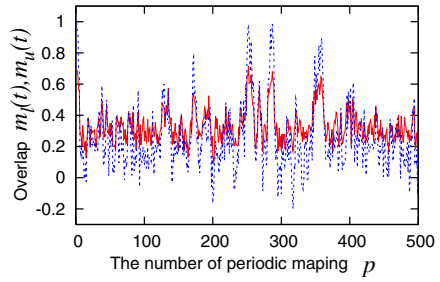


Fig. 8. The overlap $m_u(t)$ (Red solid line) and $m_l(t)$ (Green broken line) along time axis. The horizontal axis represents the p -th K -step, where $t = Kp + t_0$ ($K = 6$ and $t_0 = 1200$).

$$z_i(t + 1) = \text{sgn} \left(\sum_{j \in G_1(r_i)} [W_{ij}^{l-l} z_j(t) + W_{ij}^{i-l} y_j(t)] \right), \tag{8}$$

where u means upper layer, i , intermediate layer, and l , lower layer, respectively. W_{ij}^{u-u} is the connection weight from neuron x_j of the upper layer to neuron x_i of the upper layer, W_{ij}^{i-i} , W_{ij}^{u-i} , and W_{ij}^{i-l} , and W_{ij}^{l-l} are defined similarly.

4 Designing Attractors for Control

4.1 Motion Functions

The robot has local coordinates so that at any time t , in the local coordinates the robot is at the origin with orientation 0, as shown in Fig 9.

The robot has two driving wheels and one castor wheel, and when the two driving wheels rotate with the same velocity and reverse direction, the rotation radius can be regarded as zero if we do not consider the slippage of the wheels. Therefore, the motion of the robot at each step includes two actions. First, the robot rotates with an angle $\theta(t)$ around the present origin. Next, it moves forward for an distance $L(t)$. The two-dimensional motions of the robot can be described by two time-dependent variables $\theta(t)$ and $L(t)$. Therefore, in order to realize 2-dimensional motion of the robot using dynamical behavior of the neural network, the four hundred dimensional state pattern $\mathbf{S}(t)$ of the neural network is transformed into the rotation angle $\theta(\mathbf{S}(t))$ and the movement distance $L(\mathbf{S}(t))$ by simple coding functions. The coding functions are called *motion functions* and defined by

$$\theta(\mathbf{S}(t)) = \tan^{-1} \frac{f_y(\mathbf{S}(t))}{f_x(\mathbf{S}(t))} \tag{9}$$

$$L(\mathbf{S}(t)) = \pi d \sqrt{f_x^2(\mathbf{S}(t)) + f_y^2(\mathbf{S}(t))} \tag{10}$$

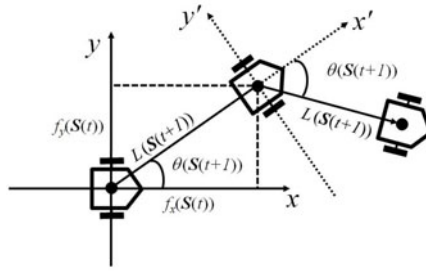


Fig. 9. The motion of the robot: at each new position, the robot has local coordinates in which the positive x axis is always in the direction toward the front of the robot

where d is the diameter of the driving wheels, and $f_x(S(t))$, $f_y(S(t))$ are the x -axis increment and y -axis increment in the local coordinates at time t defined by

$$f_x(S(t)) = \frac{4}{N} \mathbf{A} \cdot \mathbf{C} \quad f_y(S(t)) = \frac{4}{N} \mathbf{B} \cdot \mathbf{D} \quad (11)$$

where $f_x(S(t))$ and $f_y(S(t))$ are four $N/4$ dimensional sub-space vectors of state pattern $S(t)$, which is shown in Fig. 10. The inner products of $\mathbf{A} \cdot \mathbf{B}$ and $\mathbf{C} \cdot \mathbf{D}$ are normalized by $4/N = 100$, so $f_x(S(t))$ and $f_y(S(t))$ ranges from -1 to $+1$. Therefore, the rotation angle $\theta(S(t))$ takes value from $-\pi$ to π , and the movement distance $L(S(t))$, takes values from 0 to $\sqrt{2}\pi d$.

4.2 Attractors for Control Mechanisms

Upper Layer. The upper layer for sensing consists of 5 independent sub-space vectors that correspond to 5 sensors — four microphones for detecting target direction and one pair of ultrasonic sensors for detecting obstacles, shown in Fig. 10. Attractors are embedded in the case that the direction of the maximum signal intensity received by microphones is front, back, right, or left without obstacles, respectively. As the robot moves, the signal intensity of the four microphones changes. The firing state of the sensing neurons in the upper layer sensitively responds to the external signal input and produce adaptive dynamics which act on the driving neurons. On the other hand, if there are obstacles to prevent the robot from moving forward, sensing neurons corresponding to ultrasonic sensors are activated, and cause strong chaotic dynamics in the sensing neurons to act on the driving neurons so as to enable the robot to perform complex motions.

Lower Layer. The lower layer for driving consists of four groups of attractor patterns, shown in Fig. 11. Each group of patterns are embedded as a cyclic attractor. Each cyclic attractor corresponds to a prototypical simple motion. Each attractor pattern consists of four random sub-space vectors \mathbf{A} , \mathbf{B} , \mathbf{C} and \mathbf{D} . \mathbf{A} and \mathbf{B} are independent random patterns. A group of specified intra-pattern structure is given, such as $\mathbf{A} = \mathbf{C}$ or $-\mathbf{C}$ and $\mathbf{B} = \mathbf{D}$ or $-\mathbf{D}$ in the present study. By the coding of motion functions, each group of attractor patterns corresponds to a stationary motion in two-dimensional space.

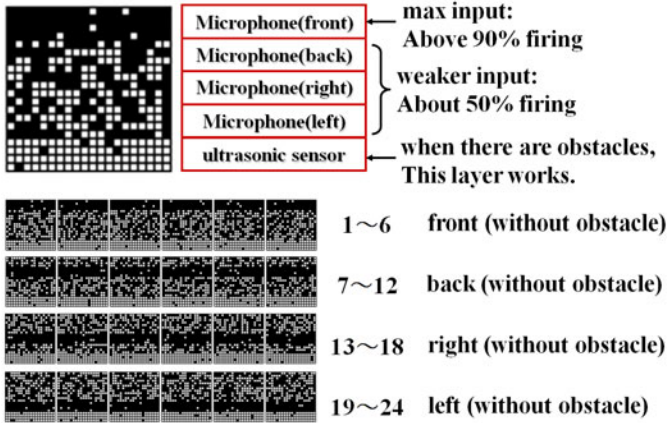


Fig. 10. Four cyclic attractors embedded in upper layer

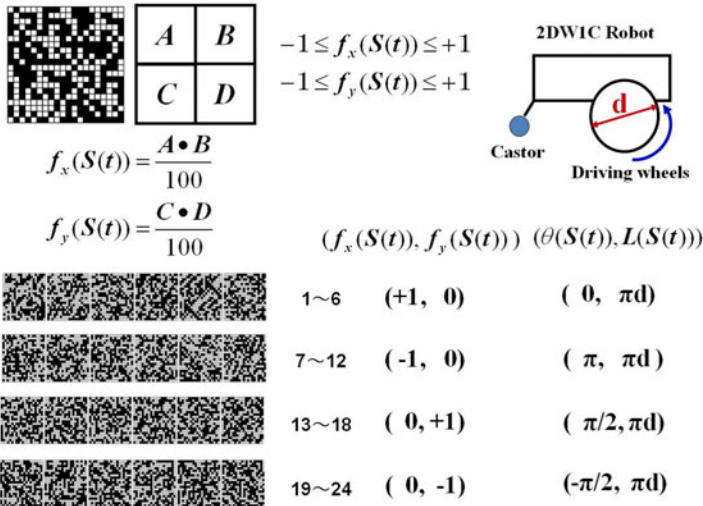


Fig. 11. Attractor patterns designed for motion control. Each cyclic memory corresponds to a simple motion.

5 Control Algorithms in Roving Robot

Studies of RNNM have shown that sufficiently large or small values of connectivity r enable the neural network to generate either chaotic dynamics or attractor dynamics [4], respectively. In a quasi-layered RNNM, a set of connectivity values ($r_{u,u}, r_{u,l}, r_{l,l}$) should be considered. In our example, when $r_{l,l}$ is set as 60, different values of $r_{u,u}$ cause chaotic dynamics with different dynamical properties. Correspondingly, with the coding of motion functions, the robot can show either weak (localized) chaotic motions (Fig 12(left)), or strong chaotic motions (Fig 12(right)).

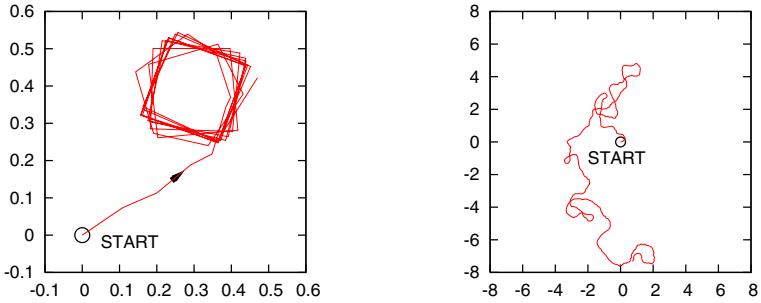


Fig. 12. Examples of motion control ($r_{u,l} = 400$ and $r_{l,l} = 60$): (left) larger $r_{u,u} = 60$ generates weak chaotic dynamics with localized property, (right) smaller $r_{u,u} = 20$ causes strong chaotic dynamics

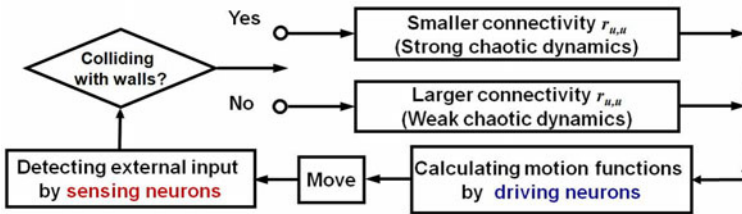


Fig. 13. Control algorithm for solving 2-dimensional mazes

Once optimal connectivities are determined, the control for motion in a two-dimensional maze will work well even when obstacle information is just given to the obstacle sub-space in the upper sensing layer. However, in the initial implementation, the obstacle information is used to adaptively switch the connectivity $r_{u,u}$ in the upper sensing layer. The control algorithm is shown in Fig. 13

The sensing neurons in the upper layer sensitively response to the intensity of the sound signal from the target detected by the four microphones. Driving neurons in the lower layer sensitively respond to the sensing neurons and the robot adaptively turns toward the strongest intensity direction quickly due to the sensitivity of the chaotic dynamics. When there are no obstacles in the range of the ultrasonic sensor, the robot moves with large value of connectivity $r_{u,u}$. When there is an obstacle, it moves chaotically with small value of connectivity of $r_{u,u}$ until it avoids the obstacle. Several examples of computer experiment are shown in Fig. 14. In the near future, switching of connectivity $r_{u,u}$ will be replaced with only sensing neurons responding to external input adaptively.

6 Experiments with Hardware Implementations

Experiments were down with a hardware implementation of the robot control system. Several kinds of typical 2-dimensional mazes were constructed. A loud speaker was set as a target, which is emitting a specific sound signal resembling the calling song of

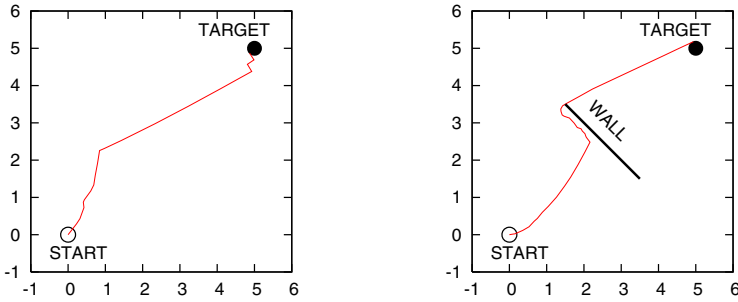


Fig. 14. Examples of computer experiments of obstacle avoidance: (left) No obstacle, r_{uu} stays at large value; (right) When an obstacle prevents the robot from moving forward, small value of r_{uu} is used to update the network

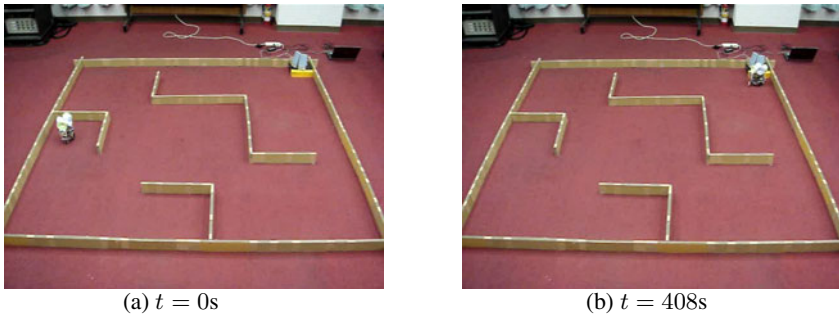


Fig. 15. Video snapshots of the roving robot walking on a horizontal floor where there are obstacles between the robot and the target. (a) Around the starting point, the robot is far from the target, and there are obstacles between the robot and the target (b) after some time of wandering, including chaotic dynamics, the robot reaches the target.

a male cricket. When the robot, like a female cricket, moves in the two-dimensional space, it can obtain only rough directional information from the four microphones, which are attached to the front, the back, the left and the right of the robot. The robot was controlled according to the control algorithm shown in Fig. 13. Using chaotic dynamics, the robot successfully avoids obstacles and reaches the target. Fig. 15 shows some video snapshots of the robot solving the above 2-dimensional maze. Now we are implementing control using input of obstacle information to sensing neurons, and will report experiments on this in the near future.

7 Developing Hardware Implementations and Pseudo-neuron Devices

Now, we briefly show our preliminary results about further hardware developments for implementations based on our ideas for functional chaos.

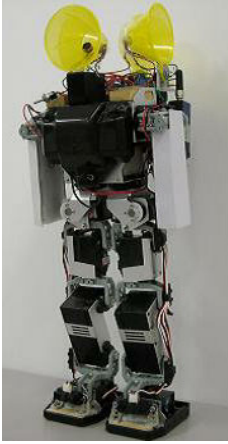


Fig. 16. A humanoid robot used in our experiments



Fig. 17. A humanoid robot executing the action of climbing upstairs

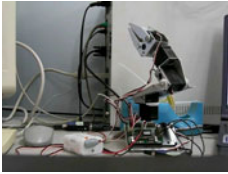


Fig. 18. An arm robot used in our experiments

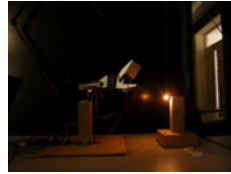


Fig. 19. An arm robot executing the action of reaching for a target

7.1 Roving Humanoid Robot and Arm Robot Driven by Chaos

In this section we introduce two examples of implemented robot systems. One applies the same control method for solving mazes to a humanoid robot with two legs, as shown in Fig. 16. This experiment is designed to extend our idea to deal with 3-dimensional configuration of obstacles, instead of 2-dimensional mazes as in the previous examples. The robot behavior necessarily includes the actions not only of straddling or stepping over obstacles but also of climbing up and down obstacles, depending on their size (see Fig. 17). This experiment is still underway.

The other experimental system applies the same control idea to arm motions of animals like human beings or monkeys. We consider the situation in which the robot does not have advanced visual information processing ability like mammals but has only a poor visual sensing system. In this situation, the sensors can detect only the rough direction of target, with some uncertainty. Four infrared light sensors are attached on the arms to realize such "ill-posed" control situations. (see Fig. 18) At present, the experiment has only been successful in the case without obstacles (see Fig. 19).

7.2 A Pseudo-neuron Device and Diffusively Coupled Network

In this section, we report our study about a pseudo-neuron device fabricated using Self Electro-optic Effect Device (SEED) and coupled dynamic SEED (D-SEED).

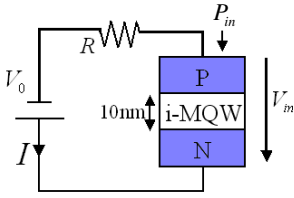


Fig. 20. single SEED

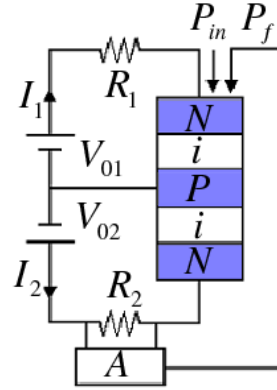


Fig. 21. Serially connected SEEDs with feedback

Self Electro-optic Effect Device (SEED) & Dynamic SEED. A typical single SEED composed of *p-i-n* semiconductors is shown in Fig. 20. The primary variable of this system is photocarrier density n which is generated by the incident optical power P_{in} . The rate equation for photocarrier density n is presented below. The important point is that it exhibits a bistable property with respect to incident optical power.

We propose a pseudo-neuron device that consists of two bistable SEED elements optically connected in a series with feedback from the lower element to the power of an incident light beam (Fig. 21). We called it "Dynamic SEED (D-SEED)".

Coupled rate equations for photocarrier density in the two SEED sections, n_1 (upper) and n_2 (lower), are written as

$$\frac{dn_1}{dt} = -\frac{n_1}{\tau_1} + \frac{\alpha_1 (P_{in} + P_f) \Omega_{01}}{\{\omega - \xi_1\}^2 + \left(\frac{\Omega_{01}}{2}\right)^2} \tag{12}$$

$$\frac{dn_2}{dt} = -\frac{n_2}{\tau_2} + \frac{\alpha_2 (P_{in} + P_f - m_1 n_1) \Omega_{02}}{\{\omega - \xi_2\}^2 + \left(\frac{\Omega_{02}}{2}\right)^2} \tag{13}$$

$$\xi_i = \omega_{0i} - \beta (V_{0i} - R_i I_i) \quad (i = 1, 2) \tag{14}$$

$$I_i = \eta_i n_i \quad (i = 1, 2) \tag{15}$$

$$P_f = A I_2 R_2 = A R_2 \eta_2 n_2 \tag{16}$$

where τ , α , Ω_0 , ω , ω_0 , β , V_{in} , V_0 , R , η are system parameters. Note that the primary parameter is P_{in} , which causes many kinds of bifurcation phenomena. m is absorption parameter and A is feedback gain parameter.

The rate equations show that for the upper SEED, feedback light is added to incident light, whereas the lower SEED receives the light decreased by the absorption in the

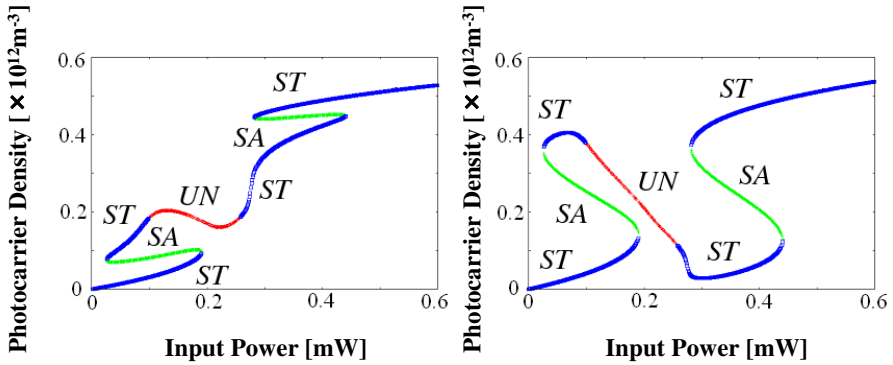


Fig. 22. Stationary solutions of upper SEED (left) and lower SEED (right) as a function of input light power P_{in} . Blue lines, green lines and red lines indicate stable "ST", saddle "SD" and unstable "UN" solutions respectively.

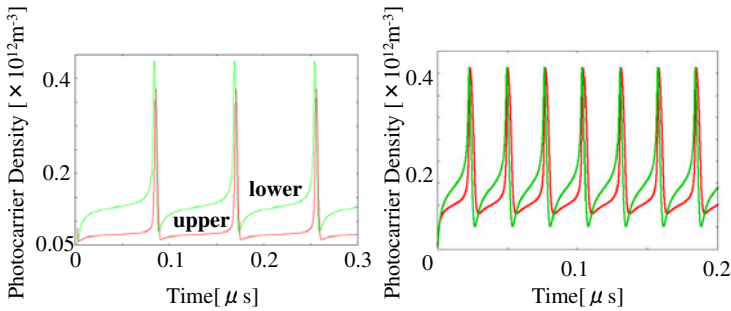


Fig. 23. Oscillatory solutions of n_1 and n_2 for input light power, $P_{in} = 0.170$ (left), 0.220 (right), respectively. Note that oscillation periods become infinitely long as the solutions approach the saddle-node bifurcation points at the edge of the unstable regions (Fig. 22).

upper SEED. When we choose appropriate parameter values, we can obtain various bifurcation phenomena as shown in Fig. 22. Particularly, important bifurcations are "Hopf bifurcation" and "Saddle-node bifurcation". [16]

The important point of this case is that a saddle-node bifurcation occurs on a marginal limit cycle, so that, the period of the limit cycle becomes infinitely long as light power approaches the saddle-node bifurcation point (Fig. 23). Two typical cases of oscillation are shown in Fig. 23.

Now we extend our idea to a network of D-SEEDs which are diffusively coupled in a two-dimensional array. The coupled rate equations can be modified to include a diffusion term, from Eq. (17).

$$D \left[\frac{\partial^2 n}{\partial x^2} + \frac{\partial^2 n}{\partial y^2} \right] \cong D (n_{i-1,j} + n_{i+1,j} + n_{i,j-1} + n_{i,j+1} - 4n_{i,j}) \tag{17}$$

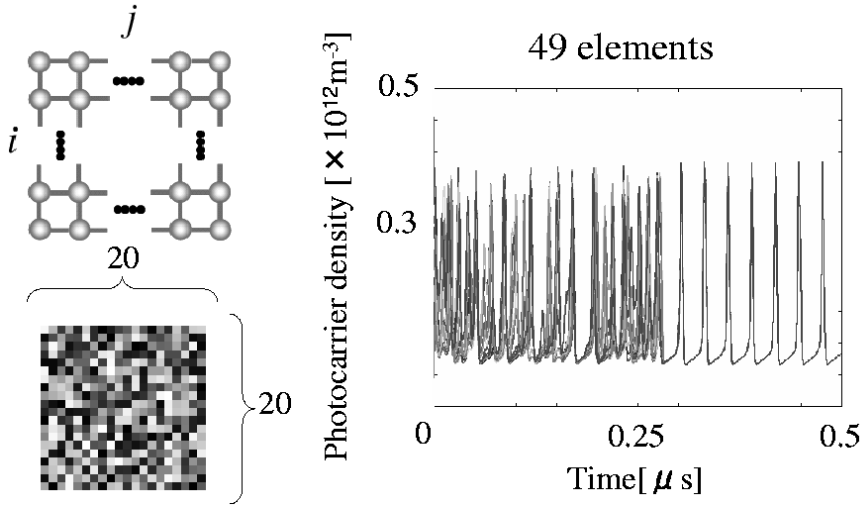


Fig. 24. Example of D-SEED network. (upper-left) Two-dimensional square lattice D-SEEDs, (lower-left) snapshot of a network state at a certain time step, where brightness of each cell corresponds to $n_{i,j}$ and is represented in grayscale normalized by maximum value of carrier density, (right) example of time-dependence of carrier density (chaotic oscillation and synchronization).

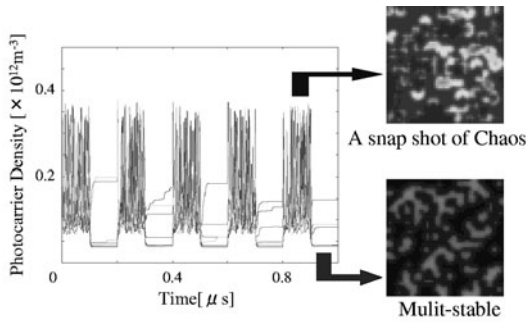


Fig. 25. Switching between two states by varying light power between $P_{in}=0.160$ (Chaos) and 0.110 (Multi-stable)

Here (i, j) means the position of a D-SEED in an square lattice of points in two-dimensions, as shown in Fig. 24 (left-up). Under certain light power, time-dependence of carrier density shows chaotic dynamics and long time behavior converges to synchronized periodic oscillation (Fig. 24 (right)). We obtained global spatio-temporal chaotic dynamics if we sufficiently increased the number of D-SEEDs. By varying light power, we also confirmed the existence of multi-stable states. In Fig. 25, we succeeded in obtaining switching between two states, chaotic state and a multi-stable state, by varying incident light power. We aim to use this property to implement control functions with adaptive switching among chaotic state, multi-stable states and synchronized periodic states.

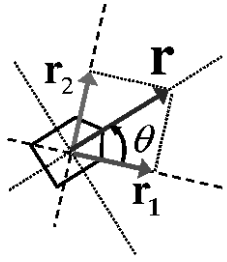


Fig. 26. 2-dimensional motion

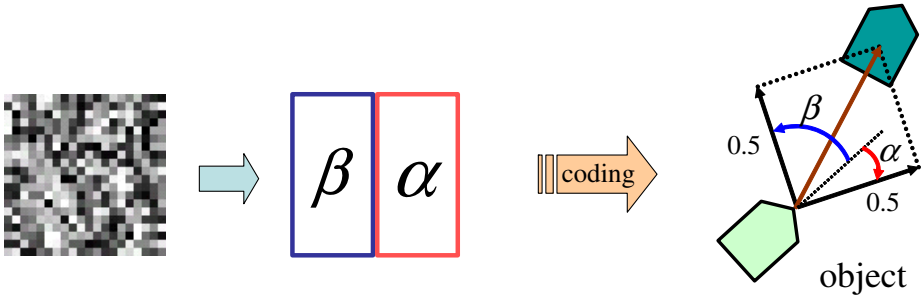


Fig. 27. Coding method (b)-1

Complex Control using Pulsed Neuron Networks. In our previous studies, the key idea was adaptive switching between chaotic dynamics and attractor dynamics in neural networks. Now, it is necessary to develop hardware implementations using artificial neuron devices. In order to apply a pulsed-neuron network such as the D-SEEDs network to the problem of solving mazes, it is necessary to determine appropriate coding functions for mapping the device states to the motion variables.

We considered two types of switching.

- (a) Adaptive switching between chaotic state and synchronized periodic state.
- (b) Adaptive switching between chaotic state and multi-stable state.

In each case, the state pattern of the network is represented by 400-dimensional state vectors, while motion in 2-dimensional space is only two-dimensional (Fig. 26). Therefore, it is necessary to convert 400-dimensional state to 2-dimensional motion by a coding function. We designed 3 different coding methods. One is shown in Fig. 27 and another is shown in Fig. 28, where the detailed definition of motion increments are omitted. An important point is that, at each sampling time for motion increments calculation, periodic or chaotic pulsed operations of photocarrier density in each element are transformed into binary states (+1 or -1) depending on being larger or smaller than a certain threshold value.

With the second method it is possible to switch between 4 monotonic motions which are almost linear motions in 4 quadrant directions and chaotic motion. Now, we can solve the maze by adaptive switching between monotonic motion and chaotic motion with a simple control algorithm shown in Fig. 29. Successful results were obtained from computer experiments and details will be reported elsewhere.

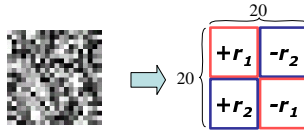


Fig. 28. Coding method (b)-2

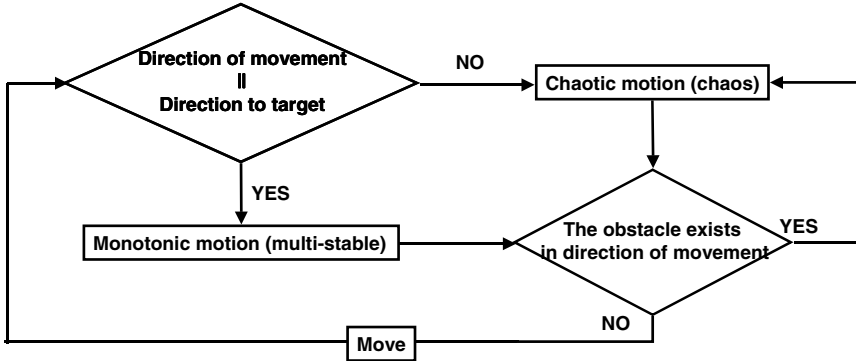


Fig. 29. Control algorithm for adaptive switching between chaotic motion and monotonic motion

8 Experiments Using Quasi-layered Recurrent Neural Network with Sensory-, Inter-, and Motor-Neurons

The proposal of the Quasi-Layered Recurrent Neural Network with three layers, composed of Sensory-, Inter-, and Motor-Neurons, is based on two aims in our heuristic research approach. One is to develop our idea about control via simple rules using chaos in a recurrent neural network model so as to enable simultaneous multi-tasks, and the other is to show phenomena of synchronization of firing patterns between far distant neurons, which are observed in physiological-, fMRI-, and EEG-experiments on brains, associated with advanced functioning (for example, see [19]). The important point of the observed synchronization phenomenon is that there does not appear to be any strongly correlated activity in the intermediate neurons located between the synchronized neurons. We consider that intermediate layers of neurons may be chaotic.

The outline of our model corresponding to Quasi-Layered Recurrent Neural Network with Sensory-, Inter-, and Motor-Neurons is shown in Fig. 31. Our computer experiments are at the stage that the same maze-solving functions as the previous models can be realized, as shown in Fig. 30. It should be noted that the connectivity's are all fixed and inputs only enter the sensory-neurons. Experiments on multi-tasks are now being done and the results will be reported elsewhere. Synchronization experiments are also underway.

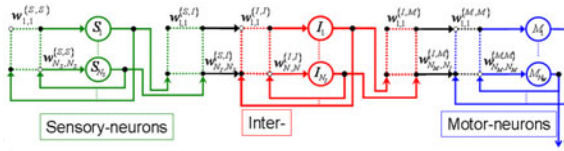


Fig. 30. A block diagram of our model for a Quasi-Layered Recurrent Neural Network with three layers, composed of Sensory-, Inter-, and Motor-Neurons

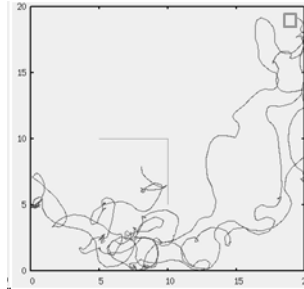


Fig. 31. Examples of successful cases of solving a maze

9 Concluding Remarks

The results of the present preliminary experiments, allow us to conclude that chaotic dynamics is useful to solve complex problems, such as mazes, not only in computer experiments, but also in hardware implementation of robot systems. Although detail considerations remain to be done, it is at least a heuristic discovery that chaotic dynamics could play important roles in biological systems. We are sure that such novel functional aspects of chaotic dynamics could be applied to complex control by simple rules in systems with large numbers of degrees of freedom, and be useful for engineering applications mimicking complex functions observed in biological systems including brain.

Acknowledgements. The authors greatly appreciate the valuable comments of Dr. Peter Davis. This work has been partly supported by Grants-in-Aid for Scientific Research, #19500191 in Japan Society for the Promotion of Science and #22120509 in the Ministry of Education, Culture, Sports, Science & Technology, also partly supported by System Development Program for Advanced Measurement and Analysis in Japan Science and Technology Agency (JST).

References

1. Skarda, C.A., Freeman, W.J.: How brains make chaos in order to make sense of the world. Behavioral and Brain Sciences 10, 161–195 (1987)
2. Huber, F., Thorson, H.: Cricket Auditory Communication. Sci. Amer. 253, 60–68 (1985)

3. Mikami, S., Nara, S.: Dynamical Responses of Chaotic Memory Dynamics to Weak Input in a Recurrent Neural Network Model. *Neural Computing & Applications* 11, 129–136 (2003)
4. Nara, S., Davis, P.: Chaotic wandering and search in a cycle memory neural network. *Progress of Theoretical Physics* 88, 845–855 (1992)
5. Nara, S., Davis, P., Kawachi, M., Totuji, H.: Memory search using complex dynamics in a recurrent neural network model. *Neural Networks* 6, 963–973 (1993)
6. Nara, S., Davis, P., Kawachi, M., Totuji, H.: Chaotic memory dynamics in a recurrent neural network with cycle memories embedded by pseudo-inverse method. *Int. J. Bifurcation and Chaos Appl. Sci. Eng.* 5, 1205–1212 (1995)
7. Nara, S., Davis, P.: Learning feature constraints in a chaotic neural memory. *Phys. Rev. E* 55, 826–830 (1997)
8. Nara, S.: Can potentially useful dynamics to solve complex problems emerge from constrained chaos and/or chaotic itinerancy? *Chaos* 13, 1110–1121 (2003)
9. Li, Y., Nara, S.: Novel Tracking Function of Moving Target Using Chaotic Dynamics in A Recurrent Neural Network Model. *Cognitive Neurodynamics* 2, 39–48 (2008)
10. Suemitsu, Y., Nara, S.: A solution for two-dimensional mazes with use of chaotic dynamics in a recurrent neural network model. *Neural Computation* 16, 1943–1957 (2004)
11. Skarda, C.A., Freeman, W.J.: *Behavioural and Brain Sciences* 10, 161–195 (1987)
12. Tsuda, I.: *Behavioral and Brain Sciences* 24, 793–847 (2001)
13. Fujii, H., Itoh, H., Aihara, K., Ichinose, N., Tsukada, M.: *Neural Networks* 9, 1303 (1996)
14. Adachi, M., Aihara, K.: *Neural Networks* 10, 83–98 (1997)
15. Nara, S., Tokuda, Y., Abe, Y., Yasukawa, M., Tsukada, N., Totuji, H.: *J. Appl. Phys.* 75(8), 3749–3755 (1994)
16. Ohkawa, Y., Yamamoto, T., Nagaya, T., Nara, S.: *Phys. Appl. Phys. Lett.* 86, 111107 (2005)
17. Suemitsu, Y., Nara, S.: *Neural Compt.* 16(9), 1943–1957 (2004)
18. Nara, S., Davis, P.: *Prog. Theor. Phys.* 88, 845–855 (1992); Nara, S.: *Chaos* 13(3), 1110–1121 (2003)
19. Yamaguchi, Y.: The Brain Computation Based on Synchronization of Nonlinear Oscillations: On Theta Rhythms in Rat Hippocampus and Human Scalp EEG. In: *Marinaro, M., Scarpetta, S., Yamaguchi, Y. (eds.) Dynamic Brain - from Neural Spikes to Behaviors. LNCS, vol. 5286, pp. 1–12. Springer, Heidelberg (2008)*

Mathematical Modeling of Human Thermoregulation: A Neurophysiological Approach to Vasoconstriction

Boris R.M. Kingma¹, Arjan J.H. Frijns², Wim H. Saris¹
Anton A. van Steenhoven², and Wouter D. van Marken Lichtenbelt¹

¹ Department of Human Biology, NUTRIM School for Nutrition
Toxicology and Metabolism of Maastricht University Medical Centre+
Universiteitssingel 50, Maastricht, The Netherlands
{B.Kingma,W.Saris,Markenlichtenbelt}@maastrichtuniversity.nl

² Department of Mechanical Engineering, Eindhoven University of Technology
Eindhoven, The Netherlands
{A.J.H.Frijns,A.A.v.Steenhoven}@tue.nl

Abstract. Skin blood flow is of major importance in human thermoregulation. Classic thermoregulation models require an explicit set point to control temperature. Normally such a set point is defined in the unit of the controlled variable (i.e. Celsius). However, the human body does not sense temperature directly, instead temperature information is coded into neuron fire rates. Here we explored the neurophysiology of thermoregulation to develop a mathematical model of skin blood flow that does not require a set point. The model was developed on measurement data of skin temperature, core temperature and skin blood flow and was validated using k-fold cross validation. The model explained over 90% of the variance in the measurements ($r^2=0.91$). Hence, the results are promising and indicate that emulation of thermoregulatory neurophysiology is able to capture the dynamics of skin blood flow control.

Keywords: Neural pathways, Skin blood flow, Temperature, Thermoreception.

1 Introduction

In order to maintain core temperature within narrow limits, the human body balances both heat gain and heat loss [1]. Conservation of body heat during mild cold challenges is primarily achieved by vasoconstriction (i.e. constriction of blood vessels), which decreases skin blood flow [2, 3]. Thereby heat transport from the core to the skin is diminished and eventually heat loss to the environment is decreased. Hence, the accuracy of models of human thermoregulation depends for a great deal on their ability to predict skin blood flow. In the past various models predicting skin perfusion responses have been developed [4]. What these models have in common is that they require an explicit setpoint; i.e. a reference temperature which is compared with the actual body temperatures to generate error signals. The effector response (in this case vasoconstriction) is assumed to be proportional to the error signal [5].

Although from an engineering perspective the meaning of a set-point might be clear, application of the concept in human physiology is still under debate as it is not clear how this set-point could be contained [6]. Alternatively, it is hypothesized that thermoregulatory effectors could also be modeled by using bell-shaped neural activation patterns of thermo-sensitive neurons, and reciprocal cross inhibition (RCI) [7-10]. An advantage of this approach is that the model structure remains true to current neurophysiologic knowledge on thermoregulation. For instance, the thermoregulatory system does not sense temperature directly, yet the information is coded into neuron fire rates [11]. Hence skin blood flow is modeled from principles of neurophysiology instead of simple regression.

In this study a mathematical model for skin blood flow during cold exposure was developed based on physiological data on neural thermo-sensitivity and neural pathways. The aim of this study was to investigate whether skin blood flow can be adequately modeled through simulation of thermo-sensitive neurons and neuro-physiological pathways of excitation and inhibition.

2 Methods

The model for the central control of skin blood flow was based on thermal reception and neural pathways that were mostly established by in vivo animal experiments. To underline the importance of modeling human physiological responses from neurophysiological principles we first address the physiological mechanisms, thereafter a mathematical translation is described.

2.1 Physiology of Vasoconstriction

Physiological experimental evidence indicates that skin blood flow is regulated by both reflex (neural) and local mechanisms [2]. Neural control of vasoconstriction is mediated by the sympathetic nervous system. Under thermoneutral conditions blood vessels are under a baseline sympathetic vasoconstrictor tone. During a cold challenge an increase in sympathetic vasoconstrictor tone causes blood vessels to constrict [3].

The ability of the body to react to a cold challenge is determined by thermal reception, neural integration of thermal information and vessel responsiveness to the increased vasoconstrictor tone.

Thermal Reception. Thermal reception is mediated through temperature sensitive neurons. The steady state fire rate vs. temperature has a characteristic bell-shaped form (see Figure 1).

In addition to steady state fire rates, temperature dynamics influence the neuron fire rate such that cold sensitive neurons will fire more often (also referred to as bursts) during cooling than during warming in the same temperature range [5, 9, 12]. Likewise a warm-sensitive neuron will fire more often when heated rather than cooled. Although there is spatial variation in the actual fire rate of neurons, the general response of individual temperature sensitive neurons has been accepted widely.

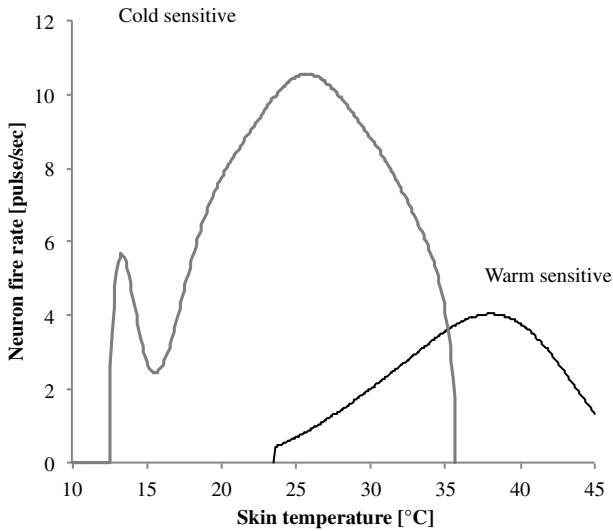


Fig. 1. Steady-state fire rate of cold sensitive neuron (gray line) and warm-sensitive neuron (black line). Adjusted from Zotterman [12]

Neural Integration. The specific integration of neural information through neural pathways is still enigmatic. However Nakamura and Morrison recently identified neural control of cold defensive responses to skin cooling in the rat [11, 13]. For the mathematical model we used their description of sensory pathways, effector pathways and related neuronal circuits (see Figure 2). Nakamura and Morrison showed that in a neutral situation, when there is virtually no cool input from the skin, cold defense pathways are inhibited by warm sensitive neurons in the hypothalamus. Hence, no vasoconstriction occurs. However, during environmental cooling, cold sensitive neurons at the skin are excited and increase their fire rate. Information of individual neurons is combined in neurons of the spinal cord where it is transmitted to the hypothalamus. There the warm sensitive neurons in the hypothalamus are inhibited, which leads to the increase of sympathetic adrenergic tone and ultimately vasoconstriction.

2.2 Modeling of Vasoconstriction

The description of thermal reception and neural integration was schematized in a diagram (see Figure 2).

In the left part of Figure 2, local skin temperatures are transduced into neural coded information by cold and warm sensitive neurons. In the spinal cord section information from individual neurons is combined and transmitted to the hypothalamus. Warm sensitive neurons in the hypothalamus transduce core temperature and are inhibited by cold sensitive neurons from the periphery, whereas peripheral warm sensitive neurons perform an excitatory role. Control neurons responsible for cold defense pathways are inhibited by the warm sensitive neurons in the hypothalamus.

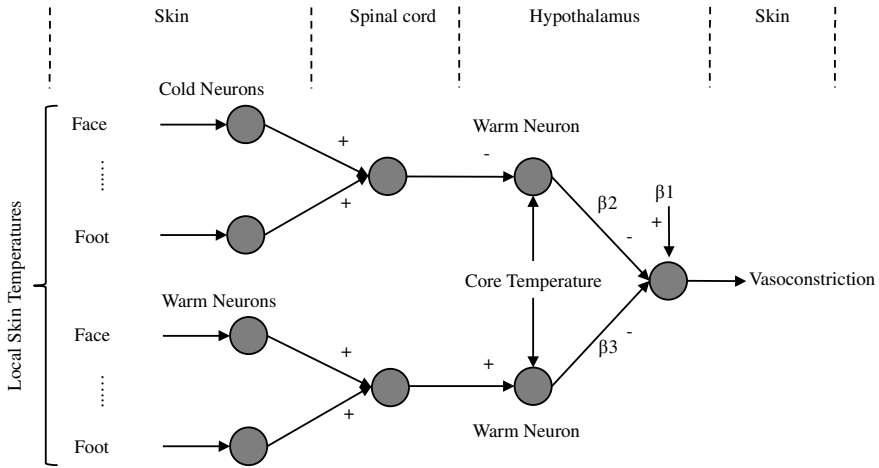


Fig. 2. Schematic of neuronal model for control of skin blood flow. + and – denote excitatory or inhibitory pathways; β_1 denotes the averaged combined effect of non-thermal inputs on skin blood flow; β_2 and β_3 denote respectively weighing of the inhibition of the cold defense pathway by warm sensitive neurons in the hypothalamus.

Table 1. Coefficients for the 10th order polynomial function of static neuron fire rate as given in Mekjavic and Morrison (1985). Coefficients in bold are corrected values.

Polynomial order	Cold sensitive neuron	Warm sensitive neuron
x_0	-0.19005313e6	0.1526647e5
x_1	0.85318078e5	-0.5147704e4
x_2	-0.16974919e5	0.7707699e3
x_3	0.19724509e4	-0.67475955e2
x_4	-0.14833377e3	0.38244284e1
x_5	0.75486723e1	-0.14664175e0
x_6	-0.26343323e0	0.38526706e-2
x_7	0.62289589e-2	-0.68496075e-4
x_8	-0.95563808e-4	0.78889647e-6
x_9	0.85949930e-6	-0.53173142e-8
x_{10}	-0.34432887e-8	0.15936041e-10

Thermal Reception. The neural input for the model is based on activation patterns of thermo-sensitive neurons on the skin and in the core region of the body. Simulation of both the static and dynamic components of thermo-sensitive neurons is based on the approach of Mekjavic and Morrison [14]. In their study they performed a polynomial fit of the static fire rate of temperature sensitive neurons (see Table 1 for the used coefficients).

Equations 1 to 3 describe the simulation of the neuron fire rate (after Mekjavic, 1985):

$$C_{i,t} = \frac{1}{\Delta t} \sum_{j=0}^{j=\Delta t-1} F_j + \left(A_0 \left(1 - e^{-j/K} \right) + P \times A \left(e^{-j/K_c} - e^{-j/K_i} \right) \right) \quad (1)$$

Here $C_{i,t}$ is the neural response at location i and time t (C for cold sensitive neurons, W for warm sensitive neurons); Δt is the time interval (60 sec); F_1 is the static neuron fire rate at $t=t-1$. A_0 and A are static gain factors that depend on the difference static fire rates between two moments in time.

$$A_0 = F_2 - F_1 \tag{2}$$

$$A = 5.0 \cdot F_1 \cdot |A_0| \tag{3}$$

$K=5.5$, $K_i=3.3$ and $K_e=5.5$ are static, inhibitory and excitatory gain factors respectively. P is a sign operator indicating an inhibitory or excitatory response. When cold sensitive neurons are heated P is negative, when the same neurons are cooled P is positive; vice versa for warm sensitive neurons. See Figure 3 for an example of neuron simulation.

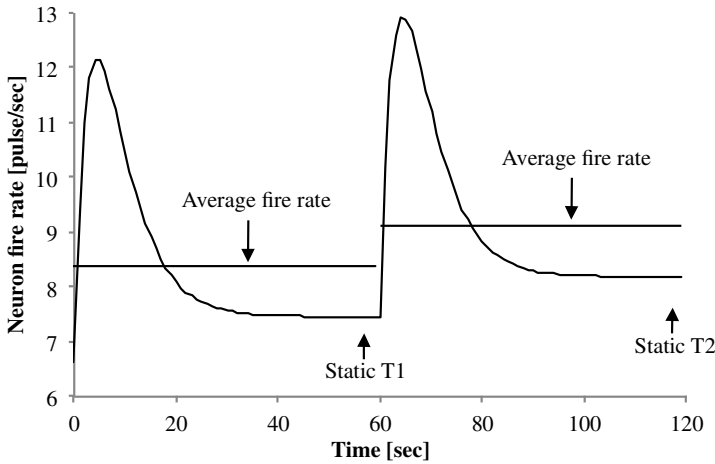


Fig. 3. Example of simulated neuron response using Equation 1. The peak indicates an excitatory response, after which the fire rate returns to its static level. The dynamic component is larger for larger variations in temperature. The average fire rate during 1 minute was used as the input fire rate for the model.

Neural Integration. As can be seen in Figure 2, neural information from skin sites was integrated at the spinal neurons. From then on, neuron response was considered as a neural drive. The resulting neural drive (N) from cold (C) and warm (W) sensitive neurons was defined as the average fire rate over all locations.

$$N_{Skin,Cold} = \frac{\sum C_{i,t}}{n_{loc}} \tag{4}$$

$$N_{Skin,Warm} = \frac{\sum W_{i,t}}{n_{loc}} \tag{5}$$

It should be pointed out that due to the non-linear characteristics of neuron fire rates, as a function of temperature and temperature history, the mean skin temperature was not used to calculate the neural drive. For example, given two temperatures $T_1=20^\circ\text{C}$ and $T_2=30^\circ\text{C}$, due to the bell-shaped form of the static neuron fire rate, the neuron fire rate of the averaged temperature (25°C) is not equal to the averaged neuron fire rates at T_1 and T_2 (see Figure 1).

The hypothalamic neural drive was calculated as the neural response to core temperature. The response of neurons in the body core is shifted by 2°C [14]. Inhibition of core neural drive by peripheral cold neurons was calculated by subtraction of cold peripheral neural drive from the core neural drive. Likewise, excitation of core neural drive by peripheral warm neurons was simulated by addition of warm peripheral neural drive on the core neural drive.

$$H_{cold} = N_{Core,Warm} - N_{Skin,Cold} \quad (6)$$

$$H_{warm} = N_{Core,Warm} + N_{Skin,Warm} \quad (7)$$

Here H denotes the net hypothalamic neural drive of either warm or cold pathway and N denotes the neural drive of neurons given their position and type.

Weighting factors for the neural drive on cold sensitive neurons (β_2 and β_3 in Figure 2) were estimated by least squares regression using the following model:

$$y = \beta_1 - \beta_2 H_{cold} - \beta_3 H_{warm} \quad (8)$$

Here y denotes the perfusion response. The constant β_1 can be interpreted as the averaged combined effect of non-thermal factors on skin blood flow.

2.3 Validation

The model is validated by k -fold cross validation. This method maximizes the available data by fitting the model on the average response of $n-1$ subjects and calculating the mean squared residuals (MSR) on the remaining subject.

$$MSR = \sum \frac{(y_t - f(x_t))^2}{n} \quad (9)$$

Where y_t is the measured perfusion at time point t , $f(x_t)$ is the model prediction at t and n is the number of measurement points in one recording. The MSR provides a measure of the quality of the model prediction, irrespective of the length of measurement. This process is iterated k times ($k=n=8$) where each fold the model is fitted and tested on a unique subset. Hence k -fold cross validation provides a measure of the ability of the model to predict the vasoconstriction response over individuals whilst maintaining the experimental conditions constant. The average MSR over k iterations is used as general measure of the capability of the model to predict perfusion.

2.4 Experimental Setup

Eight young adult males (18 to 28 years) were included (characteristics in Table 2). All subjects were healthy, non-obese and not taking medications. Subjects were in

fastened state and refrained from caffeinated or alcoholic beverages in the morning prior to the test. The medical ethical committee of Maastricht University Medical Centre+ approved the study. Each subject gave verbal and written informed consent prior to participation in the study. All procedures conformed the standards of the Declaration of Helsinki.

Table 2. Subject characteristics. Values are mean \pm standard error of the mean. (n=8).

<i>Variable</i>	<i>Mean \pm SE</i>
<i>Age, yr</i>	23.63 \pm 1.05
<i>Height, m</i>	1.81 \pm 0.02
<i>Mass, kg</i>	69.05 \pm 3.49
<i>BMI, kg/m²</i>	21.07 \pm 1.07
<i>Mean BP, mmHg</i>	85.00 \pm 1.98
<i>Whole body fat, %</i>	15.93 \pm 1.60
<i>Leisure Activity Level</i>	3.34 \pm 0.19

2.4.1 Protocol

Subjects arrived at the laboratory at 9:00 a.m. Skin temperature was measured in 1-minute intervals by i-buttons (type DS1921H; Maxim/Dallas Semiconductor Corp., USA) at the 14 positions of the ISO standard for mean skin temperature [4, 15]. Core temperature was measured in 1-minute intervals using a telemetric pill (Coretemp, USA). Whole body skin temperature was controlled by a water-perfused suit (DTI, TUBESUIT) in combination with a water temperature control unit (Blanketrol II, Cincinnati Sub-Zero). Skin perfusion was sampled at 8Hz using laser-Doppler flowmetry (Perimed, PF 5000, Sweden) at the ventral side of the hand between the base and metacarpal of the thumb. Custom made Peltier elements in the casing of the probe allowed for local temperature control. Whole body fat percentage was measured using Dual X-ray absorptiometry. Leisure activity level was indexed by a Baecke questionnaire. Subjects were in supine position and were able to watch TV. Room air temperature was kept at 24°C. A small draft in the room was allowed to assure sufficient ventilation. Before starting the measurements subjects maintained in supine position for 1 hr to become accustomed to the environment. During this period the temperature of the water suit was maintained at 33.5°C. Measurements were divided in a 15-minute baseline period where the water temperature of the suit was kept at 33.5°C followed by 15-minutes of whole body cooling where the temperature control unit was set to 10°C. Short term cooling was preferred to minimize the influence of other factors than acute sympathetic activation of the nervous system on vasoconstriction [16]. To avoid interference from local skin perfusion regulation, local skin was clamped at 33°C throughout the entire experiment [17].

2.5 Data Handling

Data handling and model development was performed using Matlab R2007a, figures were created with Microsoft Excel 2008 for Mac; statistical tests on subject characteristics were performed with SPSS16.0 for Mac. Perfusion data was resampled to 1-minute intervals using a (lowpass) FIR filter and normalized over the baseline

period. Temperature data were sampled on a minute base. Peripheral warm and cold neuron fire rates were simulated in 1-minute intervals for each measured location and sequentially averaged over subjects.

3 Results

Estimated coefficients, regression statistics and k-fold cross validation results of the neural model are presented in Table 3.

Table 3. Estimated model coefficients, β_1 : model constant; β_2 : integrated pathway of peripheral cold neurons and hypothalamic warm neurons; β_3 : integrated pathway of peripheral warm neurons and hypothalamic warm neurons. Regression statistics: p-value, r^2 and averaged mean squared residuals (MSR).

<i>Parameter</i>	<i>Value</i>
β_1	25.48
β_2	-1.44
β_3	3.26
R^2 -value	0.91
p-value	p<0.001
Averaged MSR	0.087
Averaged variance	0.080

Neural model regression analysis revealed significant fits on the measured data. Given the high r^2 -values the majority of the measured variance could be explained by the model (Table 3). Furthermore the averaged MSR of the k-fold cross validation is close to the time-averaged variance in the measurements, indicating that there is a small bias in the model prediction.

Perfusion measurements and the model prediction are shown in Figure 4. After $t=15$ subjects were cooled and vasoconstriction is observed immediately. After 10 minutes of cooling perfusion reached a nadir. The fitted line through the data points represents the prediction of the neural model.

4 Discussion

In this study a model for vasoconstriction during cold exposure was developed based on neuro-physiological concepts. Simulation of thermo-reception through warm and cold sensitive neurons was adapted from work by Mekjavic and Morrison [14]. Neural integration pathways were based on experiments by Nakamura and Morrison [11, 13]. Neural drives that were calculated by the model were fitted to human experimental skin blood flow data. Given the high value of explained variance, the model predicts vascular responses to a mild thermal cold stimulus adequately. Furthermore, the averaged MSR values are close to the variance of the measurements. Therefore, this study shows that an explicit declaration of a set point is not necessary for modeling skin perfusion during short term cooling.

4.1 Limitations

The neuron response and neural afferent pathways are established in small mammals and projected on human response. Therefore, the modeled pathways might deviate from the actual pathways in humans. However, as long as no detailed human studies on neural pathways and integration are available we have to rely on these elaborate animal studies.

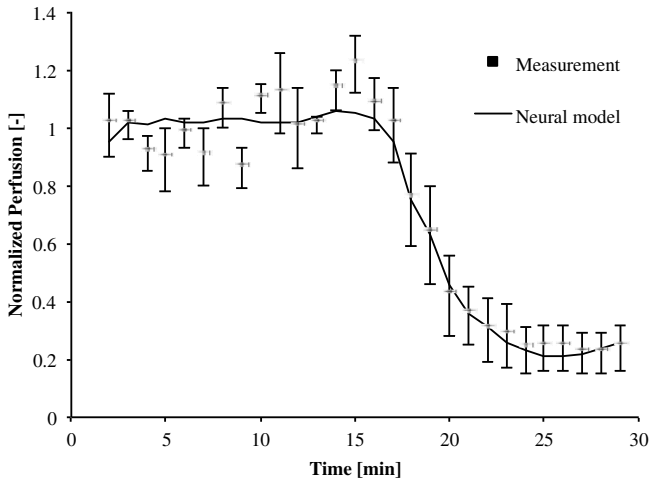


Fig. 4. Measured and fitted perfusion response. After 15 minutes whole body cooling was performed. Error bars represent SEM (n=8).

In general the thermoregulatory response is subject to both thermal factors and non-thermal factors such as blood pressure and exercise or pathologies like motion-sickness and fever [5]. The experimental set-up was developed to minimize the influence of other factors than central sympathetic regulation on vasomotor response. Short term cooling was used to avoid hormonal regulation and local skin temperature was clamped to avoid local regulation of skin blood flow.

The authors acknowledge that the current model coefficients were not validated against data sets with different experimental conditions or on different subpopulations (e.g. gender, age, adiposity). Therefore it is not possible to conclude that the coefficients hold for other types of thermal challenges or different subpopulations.

In this study we did not incorporate the effect of differences in spatial thermo-sensitivity. With a greater dataset it might be possible to assign weights to the individual branches of thermo-sensitive input (face, hand, chest, etc.). It is however not possible to use data of published studies, because usually mean skin temperatures are presented. Our model instead requires local skin temperature data.

5 Conclusions

In summary, this study presents a mathematical model for skin blood flow during cold exposure based on thermo-sensitive neurons and neurophysiological pathways. The model was fitted to experimental data where young adult males were exposed to a short mild cold exposure. The model explained over 90% of the variance in the measurements ($r^2=0.91$). Hence, although further research is warranted, the results are promising and indicate that emulation of thermoregulatory neurophysiology is able to capture the dynamics of skin blood flow control.

References

1. Hardy, J.D., Dubois, E.F.: Regulation of Heat Loss from the Human Body. *Proc. Natl. Acad. Sci. USA* 23(12), 624–631 (1937)
2. Kellogg Jr., D.L.: In vivo mechanisms of cutaneous vasodilation and vasoconstriction in humans during thermoregulatory challenges. *J. Appl. Physiol.* 100(5), 1709–1718 (2006)
3. Savage, M.V., Brengelmann, G.L.: Control of skin blood flow in the neutral zone of human body temperature regulation. *J. Appl. Physiol.* 80(4), 1249–1257 (1996)
4. Parsons, K.C. (ed.): *Human Thermal Environments*, 2nd edn. Taylor & Francis (2003)
5. Mekjavic, I.B., Eiken, O.: Contribution of thermal and nonthermal factors to the regulation of body temperature in humans. *J. Appl. Physiol.* 100(6), 2065–2072 (2006)
6. Romanovsky, A.A.: Thermoregulation: some concepts have changed. Functional architecture of the thermoregulatory system. *Am. J. Physiol. Regul. Integr. Comp. Physiol.* 292(1), R37–R46 (2007)
7. Blich, J.: A theoretical consideration of the means whereby the mammalian core temperature is defended at a null zone. *J. Appl. Physiol.* 100(4), 1332–1337 (2006)
8. Benzinger, T.H.: Heat regulation: homeostasis of central temperature in man. *Physiol. Rev.* 49(4), 671–759 (1969)
9. Hensel, H.: *Thermoreception and Temperature Regulation*. Monographs of the Physiological Society. Academic Press Inc. LTD., London (1981)
10. Hammel, H.T., et al.: Temperature Regulation by Hypothalamic Proportional Control with an Adjustable Set Point. *J. Appl. Physiol.* 18, 1146–1154 (1963)
11. Nakamura, K., Morrison, S.F.: A thermosensory pathway that controls body temperature. *Nat. Neurosci.* 11(1), 62–71 (2008)
12. Zotterman, Y.: Special senses: thermal receptors. *Annu. Rev. Physiol.* 15, 357–372 (1953)
13. Nakamura, K., Morrison, S.F.: Preoptic mechanism for cold-defensive responses to skin cooling. *J. Physiol.* 586(10), 2611–2620 (2008)
14. Mekjavic, I.B., Morrison, J.B.: A model of shivering thermogenesis based on the neurophysiology of thermoreception. *IEEE Trans. Biomed. Eng.* 32(6), 407–417 (1985)
15. van Marken Lichtenbelt, W.D., et al.: Evaluation of wireless determination of skin temperature using iButtons. *Physiol. Behav.* 88(4-5), 489–497 (2006)
16. Johnson, J.M.: Mechanisms of vasoconstriction with direct skin cooling in humans. *Am. J. Physiol. Heart Circ. Physiol.* 292(4), H1690-1 (2007)
17. Kingma, B.R., et al.: Cold-induced vasoconstriction at forearm and hand skin sites: the effect of age. *Eur. J. Appl. Physiol.* 109(5), 915–921 (2010)

Visual Target Selection Emerges from a Bio-inspired Network Topology

Wahiba Taouali, Nicolas Rougier, and Frédéric Alexandre

Université Henri Poincaré- LORIA, Campus Scientifique
Vandoeuvre-lès-Nancy Cedex, France

INRIA Nancy - Grand Est Research Center, Villers les Nancy Cedex, France
{wahiba.taouali,nicolas.rougier, frederic.alexandre}@inria.fr

Abstract. The orientation of sensors toward regions of interest of the environment is an important motor activity, monitored by ancient structures of the brainstem. Particularly, the superior colliculus is known to be deeply involved in visual saccadic behavior. Target selection relies on various hints including exogenous information about the nature and the position of candidate targets and endogenous information about current motivations. We present a model of the collicular structure based on biological data, the specificity of which is related to the homogeneity of the underlying substratum of computation. This makes it more suitable to process massive visual flows on a distributed architecture, as it could be requested in a realistic task in autonomous robotics. The present model is restricted to the exogenous part of the visual pathway, from the retina to the superior colliculus. A realistic behavior for the selection of exogenous targets is reported here.

Keywords: Superior colliculus, Dynamic neural field, Visual attention, Ocular saccades.

1 Introduction

Displaying an intelligent behavior is often synonymous of intelligently exploiting the surrounding environment. In many animals, this is massively performed through the analysis of information by the visual channel, to orient subsequent behavior (perceptual decision and action). Much research in computational intelligence aims at endowing animats with such powerful skills, drawing inspiration from the living science, at several levels of description. At the functional level, it is important to know which information animals extract from the visual input, possibly in parallel communicating processing flows and, accordingly, which representations are built and exploited in memory systems. At the physiological level, if one wishes a deep anchoring in biological inspiration, the functional behavioral analysis has to be mapped onto the neural substratum and its known anatomy and physiology, which can also give indications about the way behavioral properties can emerge from fine grained neural computations. At the operational level, a formalism of computation has to be defined to implement the corresponding models, as a compromise between accuracy to biological inspiration and efficiency of computation for animats plunged in the real world.

1.1 At the Functional Level

It has been proposed for a long time [1] that two separate visual systems can be described in the ventral (temporal) and dorsal (parietal) visual cortex. These associative cortical areas were first respectively presented as dedicated to identification and location of objects. Later [2], it was more precisely explicited that the ventral cortex elaborates the construction of the perceptive representation of objects in the world, thanks to its privileged relationships with the limbic system, center of the declarative memory, whereas the dorsal cortex, linking the primary visual cortex and the motor cortex, allows for the visual control of actions towards that objects, by extracting, in the visual flow, information useful for the preparation of actions (eg. size of an object for anticipating the size of the grip in a grasping movement).

In [3], the authors explain that both cortical axes are separate but deeply interacting in primates and conclude that this dual view reconciliates the reconstructionist approach by D. Marr, very much influential in the domain of computer vision, and the purposive-animate-behaviorist approach by J. Gibson, very popular in reactive robotics. Particularly, the interplay between both functional analysis is well illustrated by two adaptive behaviors that allow to decrease the huge amount of information brought by the visual channel: visual attention proposes a sequential processing of possible targets; saccadic movements orient the body and particularly the fovea on regions of interest in the visual scene. In both cases, the visual control of action is dedicated to elaboration of a flexible and powerful representation of the visual information. According to [4], fixation and attention can be considered as mechanical and neural deictic devices and the authors explain the computational power of such a technique for variable binding and other strategies of embodied representation. At the cerebral level, the premotor theory of attention [5] stipulates that there are common processes between these key behaviors and, more precisely, that they share common neuronal circuits: Attention would be pre-programming of a saccade.

1.2 At the Physiological Level

In the above mentioned paper [3], the authors also make a very precious reference to phylogenesis and indicate that, in more primitive animals, the goal of vision is not to see but to guide their movements. The visual control of actions corresponds to a direct instantaneous perception system, whereas the identification of objects can be seen as a relation from the present visual input to a past information stored in declarative memory, not present in ancient species (eg. reptilians, amphibians). Consequently, the general purpose network that we observe in the brain of primates, where the cortex plays a major role, must be also related to the basic visual system of a frog [6], where only several input/output sensorimotor lines (predator avoidance, prey catching, locomotion guiding) define a kind of purposive vision. The very nature of the evolution of the brain makes that these ancient visuomotor structures are still present, though modulated of course by advanced control systems and coordinated by more recent memory systems.

Particularly, these circuits converge, in the frog, in a neural structure called the tectum, directly linking retinal inputs to motor actions. In mammals, the similar structure is

called the superior colliculus (SC). Indeed, this small structure in the midbrain of mammals is known to be implicated in these sensorimotor behaviors. From an hodological viewpoint, it integrates visual information from many sources (cortical or not) in the brain and sends projections toward the brainstem premotor circuits that trigger saccades [7]. From an anatomical viewpoint, it consists of a set of topological maps, mapping the surrounding space, from visual to motor reference frames [8]. And from a physiological viewpoint, its inactivation or electrical stimulation confirms its role in visual attention and saccades [9].

Many models have studied the SC and associated properties (cf. [8] for a review). We just mention here some models underlying the link to information flows and underlying behavior. The structure of the model described in [10] underlines that the main task is to decide when and where the saccade must be performed. As a consequence, two hierarchical axes are defined. The When axis (corresponding to the FEF (Frontal Eye Field) area in the prefrontal cortex) decides when to leave the current fixation point, whereas the Where axis corresponds to the SC and implements a spatial competition between candidate targets. A double-axis model combining FEF and the SC is also proposed in [11], to explain the integration of exogenous elements (external stimuli coming from the retina to the SC) and endogenous elements (internal expectancies or instructions elaborated in the prefrontal cortex). Later on, [12] proposed a competitive integration model based on strong experimental evidences at the behavioral level, indicating that all these elements (spatial vs temporal processing and integration of exogenous vs endogenous stimuli) can be integrated in a unique map, seen as a model of the SC. This common saccade map also includes features generally reported as physiologically plausible in the SC: a local excitation in the map allowing to combine close stimuli and a wider inhibition mechanism to trigger a competition between far stimuli. This interaction scheme explains why it was possible to use such a formalism as Dynamic Neural Field (DNF) [13] to implement this kind of model, as it is also the case in [14,15].

1.3 At the Operational Level

Many recent models of the SC explain a wider range of visuomotor and more generally cognitive functions at the price of a more complex internal circuitry describing the SC. Indeed, those models define several kinds of units, depending on their location on the map, which is not very consistent with the principle of homogeneity in DNF. More precisely, in [14], the reported behavior is obtained with some units standing for the currently fixated stimulus (consequently in the fovea), other units representing potentially fixated stimuli in the periphery, both kinds sending inhibition to other units triggering saccades toward a target. These kinds of units, also exploited in the model by [15], are presented as representing respectively so-called fixation, build-up and burst neurons, which are sometimes reported as parts of the intermediate layer of the SC [16], though this is still to be clearly established. In [12] also, the substratum of computation is not homogeneous, since the sensitivity of units decreases with their eccentricity onto the map. This trick is used to reproduce the observation that the latency of a saccade toward a target, presented together with a distractor, is longer when the distractor is closer from the rostral zone of the SC (corresponding to the fovea).

Consequently, these complex models often rely on physiological considerations that are controversial and their inner mechanisms can often be described as ad hoc, designed to stick to experimental observations. Moreover, this additional complexity is often obtained by introducing numerous parameters, which affects negatively the robustness of these models. Also, it becomes difficult to simulate on-line the analysis of a visual flow, due to the amount of generated computations. Such an assessment is contradictory to the ordinary view that neuronal structures are often homogeneous, due to the repetitive tiling of elementary circuits of neurons. It is also contradictory to the spirit of DNF that have been designed as a generic homogeneous model for such populations of neurons. For that reasons, in this paper, we present a model of the SC, based on DNF formalism, with an identical functioning rule for all the units in the map. Moreover, obtaining such an homogeneous substratum can yield fully distributed computation, which is important to design models that can be used online in robotic visuomotor tasks. Also, this approach has a common ground with the reactive and enactive frameworks mentioned above, which state that fundamental properties can emerge from low-level, basic computations and not from a high-level structured module. Correspondingly, one of our questions here is to observe which of the known properties of the SC we can obtain with such an homogeneous and simple brick of computation.

Finally, the proposed model is not an isolated structure but is a function of exogenous information flows and associated geometrical properties. We have explained that the SC is undoubtedly an important integrative structure, to be included in a cognitive neuroscience modeling approach of visuospatial behaviors and we will illustrate accordingly how this model could be exploited in more high level tasks.

2 Model

The computational paradigm supporting the model is grounded on the notion of a unit that is essentially a set of time dependent values varying under the influence of other units via learnable weighted links (fig. 1). The evolution of units' value is defined by a set of differential equations expressed in standard mathematical notations. The units are organized into groups that form a network and each unit can be linked to any other unit (including itself) using a weighted link. The modeling framework¹ offers a set of core objects needed to design and run such networks. However, in this framework, what is actually computed by a unit and what is learnt are the responsibility of the modeler who is in charge of providing the equations governing the unit's behavior and the plasticity of its links. Such a modeling framework is actually strongly constrained and cannot cope for example with standard artificial neural networks. It is indeed centered around a set of four principles (distributed, asynchronous, numerical and adaptive) that we think may help to bring insights on our understanding of computational intelligence. While many computational models involve explicit symbols and/or a central supervisor, this framework is able to guarantee to a certain extent the absence of such artifacts. In the end, what is achieved by such a model is the sole result of the interaction of many units working together.

¹ See <http://dana.loria.fr>

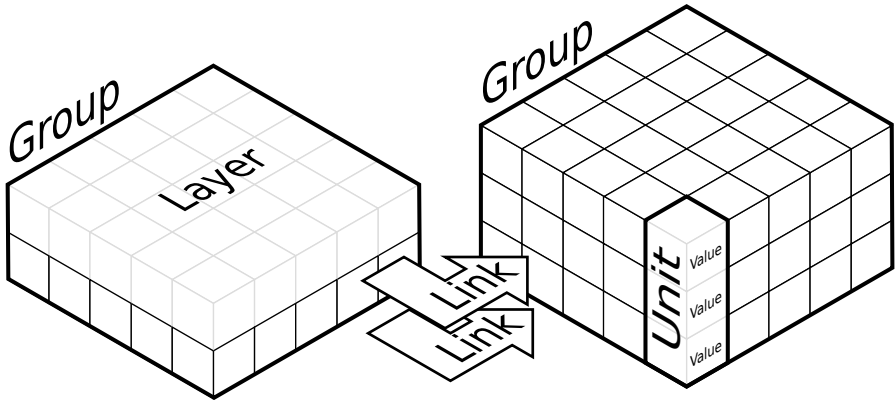


Fig. 1. A unit is a set of one to several values (V_i). A group is a structured set of one to several homogeneous units. A layer is a subset of a group restricted to a unique value V_i . A layer is a group. A link is a weighted connection between a source group to a target group. A group can be linked to any other group including itself.

2.1 Model Architecture

In short, we consider here projections to the SC coming from the retina and the primary visual cortex, carrying exogenous information, and not those coming from the frontal cortex, carrying endogenous information. Under that restriction, we want to check to what extent the model exhibits some of the well known properties of saccadic behavior associated to the SC. More specifically, our goals are to analyze the topology of the visual information that we obtain after applying this very simple transformation, together with the associated behavioral properties.

Consequently, the model is made of three distinct groups (see fig. 2) modeling the visual pathway from the retina (R) to the superior colliculus (SC) through the primary visual cortex (V1):

- retina (R, 256×512 units) receives visual input from a CCD camera.
- visual cortex (V1, 256×256 units) implements the actual cortical magnification.
- superior colliculus (SC, 63×63 units) is the place where salient locations enters competition.

The retina model is restricted to the right visual field as it is known to be the case in mammals visual pathway (left visual field projects to right colliculus and right visual field projects to left colliculus). We used an image size of 512×512 pixels and fed the retina with a normalized gray-level image of size 512×256 pixels. We will now detail the cortical magnification occurring between the retina and the primary visual area V1 as well as the competition occurring within the superior colliculus resulting in a unique localized packet of excitation designating the selected target.

2.2 Cortical Magnification

The retina represents the sensory input space and possesses a complex structure composed of several layers of neurons. Vision actually starts early in the layer of

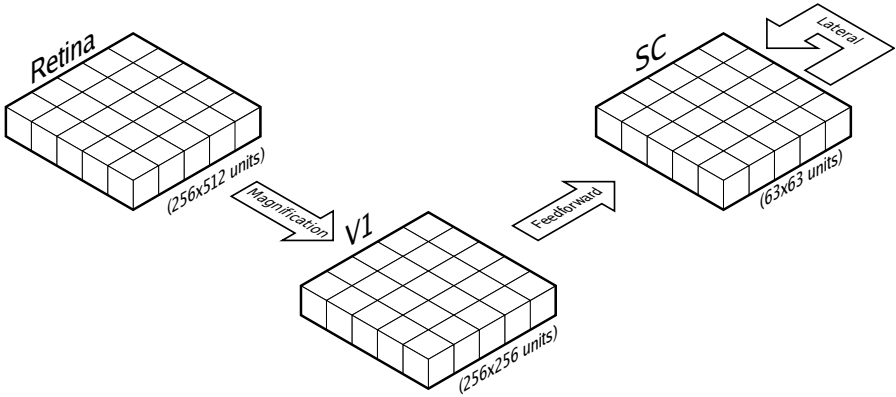


Fig. 2. The model is made of three distinct groups. The retina receives input from a CCD camera and transmit information to the primary visual cortex where the actual magnification occurs. This result is then feed to the superior colliculus where salient locations enter competition.

photo-receptors from where the flow of information is processed via the ganglion cells which are large nerve cells whose cylindraxes form the optic nerve. Due to the non-homogeneous repartition of photo-receptors on the (human) retina surface, visual acuity decreases from the center of the retina (fovea) to its periphery. This property is attributed to a variation in the density of photo-receptors that decreases from the center to the periphery [17]. Consequently, the foveal region benefits from a much higher resolution than peripheral regions and this property is preserved along the visual pathway up to early visual areas [18]. This is referred to as *cortical magnification*. To analyze this magnification in a quantitative way, a coordinate system is often defined in the visual field. The coordinate system that is best suited to the visual system is the polar coordinates (ρ, ϕ) . It characterizes a position in the visual field by its eccentricity ρ from the center of gaze and its polar angle ϕ is measured, for example, in relation to the lower vertical meridian. We can therefore define a retinotopic map which corresponds to the spatial transformation of the image by the spatial arrangement of the grid of neurons. It is often approximated by a log-polar transformation of the spherical image centered on the eye [19]. We used a simplified model of the retina considering only the photo-receptors layer. And for computational reasons (speed), we did not enforce the non-uniform repartition of photo-receptors on the retina surface. Instead, we modeled a uniform distribution of neurons onto the retina associated with a deformed polar coordinate system as proposed by [20]. Each cortical visual cell is supposed to be connected to a single or several photo-receptor cells, with respect to a logpolar deformation, that form its receptive field. So the non uniformity is caused by the changing size of the receptive fields. We used equations mapping retinotopic polar coordinates (ρ, ϕ) onto V1 Cartesian coordinates (x, y) . These equations were first introduced by [20]:

$$x = B_x \ln \left(\frac{\sqrt{\rho^2 + 2A\rho |\cos(\phi)| + A^2}}{A} \right) \tag{1}$$

$$y = B_y \arctan \left(\frac{\rho \sin(\phi)}{\rho |\cos(\phi)| + A} \right) \tag{2}$$

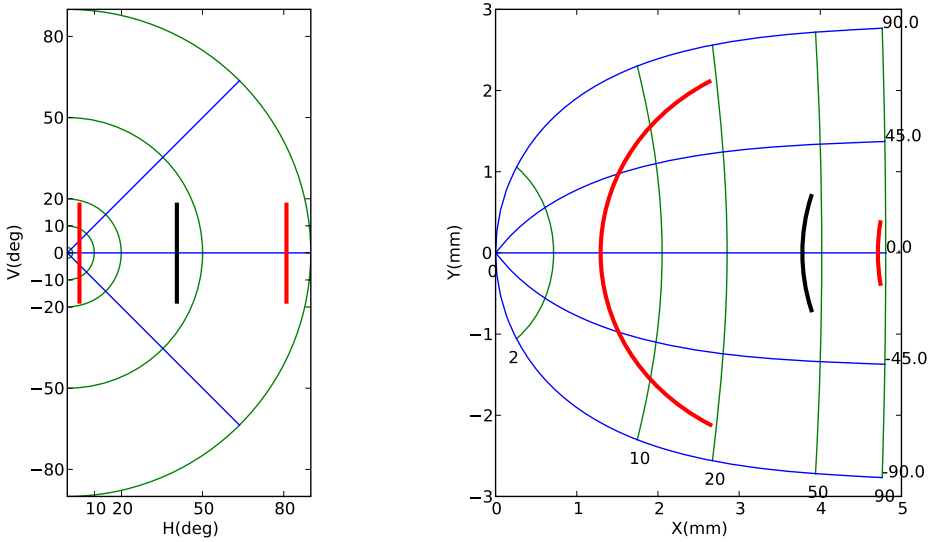


Fig. 3. Cortical magnification from the retina to the visual cortex distorts geometrical properties of the image while keeping neighborhood relationship

with $A = 3^\circ$, $B_x = 1.4mm$, $B_y = 1.8mm$. These parameters have been chosen to fit the stimulation map of the SC given by [19]. A neuron in the visual cortex fires an action potential when a visual stimulus appears within its receptive field. But for any given neuron, it may respond best to a subset of stimuli within its receptive field corresponding to its preferred direction. Neurons with similar tuning properties (what the neurons respond to) tend to cluster together but the exact structure is still unclear. Then, it is acceptable to assume that V1 has a retinotopic map similar to the collicular motor map in [21]. It means that a cell at a given position (x, y) in the V1 map is activated by retinal cells in positions (ρ, ϕ) according to given equations. One result of this deformation is that the same stimulus causes a large activation in the V1 map if it is located near the fovea and smaller activation in peripheral positions (cf. figure 3). Visual receptors of V1 have been modeled in two dimensions corresponding to an eye visual hemifield with no connection between the different receptors.

2.3 Dynamic Neural Field Theory

Collicular population (the motor layer of one superior colliculus) has been modeled with respect to the dynamical neural field theory [22][13][23] that describes the evolution of a neural population using equation (see [24] for details):

$$\tau \frac{\partial u(\mathbf{x}, t)}{\partial t} = -u(\mathbf{x}, t) + \int w(\mathbf{x} - \mathbf{y})f(u(\mathbf{y}))d\mathbf{y} + h + I(\mathbf{x}, t) \tag{3}$$

where \mathbf{x} denotes a location onto the SC; t is time; $u(\mathbf{x}, t)$ denotes the membrane potential of a neural population at point \mathbf{x} and time t ; τ is the temporal decay of synapses, f is a sigmoid function computing the mean firing rate, w is a neighborhood function, $s(\mathbf{x})$

is the input received at position \mathbf{x} and h is the mean neuron threshold. w has been set as a difference of Gaussian (*DoG*) with short-range excitations and long range inhibitions following anatomical and physiological data as reported in [25]:

$$w(\mathbf{x} - \mathbf{y}) = Ae^{-\frac{|\mathbf{x} - \mathbf{y}|^2}{a^2}} - Be^{-\frac{|\mathbf{x} - \mathbf{y}|^2}{b^2}} \quad (4)$$

and f has been set as a simple rectification of \mathbf{x} . The input $I(\mathbf{x}, t)$ is a direct one-to-one relationship according to V1 and SC respective sizes.

3 Results

The reported experimental results are of three kinds. Firstly, we check that basic properties of information encoding are ensured (topology, accuracy). Secondly, we examine the resulting saccadic behavior for target selection from exogenous information, particularly depending on the position of candidate targets with regard to the fovea. Thirdly, we address more difficult cases, particularly considering natural images and introducing the need for endogenous information.

3.1 Output Decoding

One of the questions related to the superior colliculus concerns the proper way to decode the output. Since the amplitude and direction of a saccade depend on the activity of the neural population in the deep SC [26], different ways of SC output evaluation have been proposed in the past:

- winner-take-all where the most active site indicates the direction
- summation [27,28] where all activities of active neurons are summed with weights determined by their individual labels
- weighted average [29] using a normalization according to the number of active neurons

These three evaluation schemes are equivalent in the case of a normally activated population but differ when there is a deactivation or an over-activation of a part of the population. We have retained the last decoding scheme because the superior colliculus was modeled using a dynamic neural field and it is thus ensured that a stereotyped activity profile emerges anytime corresponding to the most salient location of the V1 area. Furthermore, this stereotyped activity possesses a Gaussian shaped two-dimensional profile and it is possible to find its center of mass. We have been testing the accuracy of this coding scheme by feeding the model with standard Gaussian shaped stimuli at different locations (see figure 4). Despite the magnification effect, one can see that the model has a high precision in the standard saccadic range (-30° to $+30^\circ$, 0 to 50). We also tested the inactivation of a subpart of the collicular layer to check that we obtain both hypometric and hypermetric saccades as reported in [19] (results not presented here).

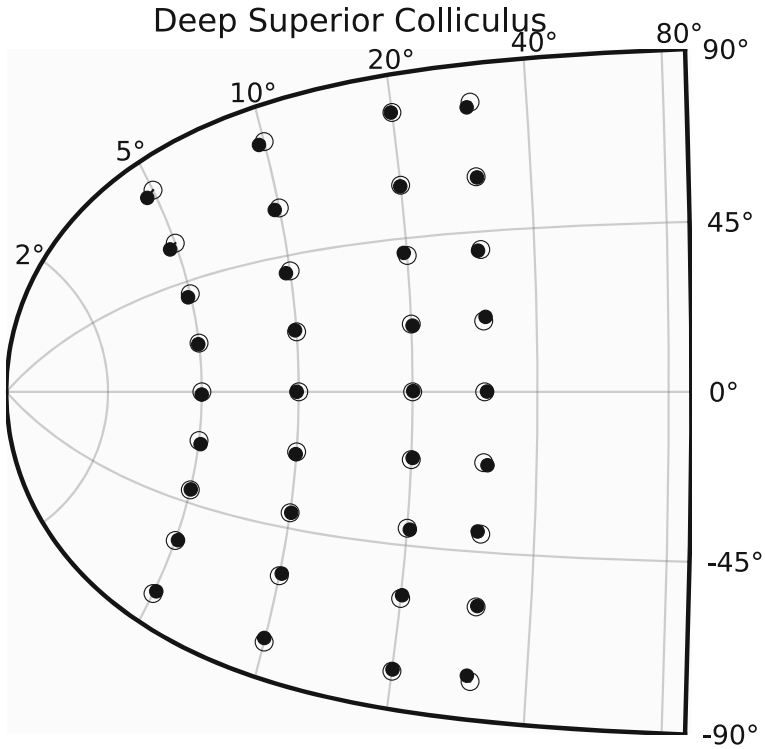


Fig. 4. Accuracy of the model of the superior colliculus has been measured using a set of retina targets that have been sequentially presented to the SC model. For each target and after convergence (difference of activity between time t and time $t + dt$ is negligible), the center of mass of the collicular activity has been decoded and represented as a circle (black dots represent the actual projection of the target in collicular coordinates).

3.2 Target Selection from Exogenous Information

Several studies have provided data on the organization of the saccadic path [30,31,32]. A set of experiences on adults with normal vision showed that the attractive value of a visual stimulus depends strongly on its distance relative to the previous fixation point (short distances preferred). Moreover, for several targets at the same distance, this attractive value is greater when the eccentricity is less important (targets closer to the fovea preferred). This result can be interpreted in the purposive framework evoked above, associating vision and preparation for action. In this perspective, shorter saccades are preferred and a nearby object is more interesting than a distant object for example in the case of hunger or danger.

Interestingly, our model displays a similar behavior and provides an explanation that is based on the topology of the neural network preparing the saccade. On the one hand, the spatial distribution of collicular neurons and their receptive fields resulting from the log-polar transformation (cortical magnification) reflect in a qualitative way how the visual information is transformed from the retina to the motor map of the superior colliculus.

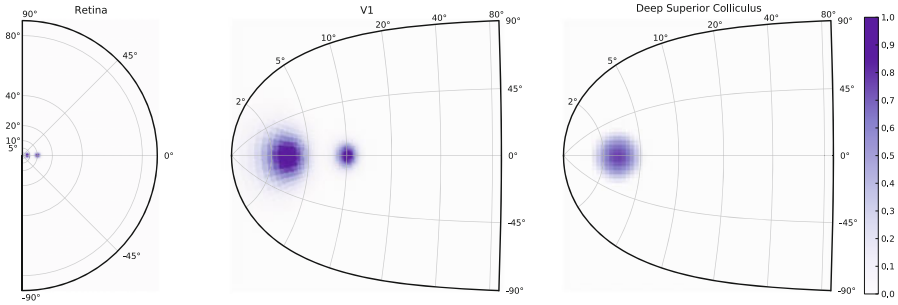


Fig. 5. The projection of two equivalent horizontal stimuli but at different eccentricities. The stimulus nearer to the fovea is automatically selected to be the saccade target.

A visual stimulus projected onto the foveal region evokes more neural activity (on the rostral part of the collicular map) than a similar stimulus in the peripheral region. On the other hand, the connectivity of the DNF model plays the role of a "winner-take-all". The profile of inhibition ensures that the system reaches a stable state once a neighborhood is recruited; there is always a selection at the end of the process. But this selection made at the premotor level does not always reflect a sensory selection: In some cases, the recruited population corresponds to an averaging and the final target may be a position where there is no stimulus; this depends on the profile of the lateral connections. Figure 5 reports an experiment where the model is tested using two punctual equivalent stimuli (same aperture, intensity and shape). Their attractive value is estimated in V1 map. The cortical population activated by the stimulus at 3° is larger than the population activated by the stimulus at 10° . Then, the resulting activity after computation in the deep layer of the superior colliculus is a stable bubble in the first position. So it can be said that the selection of the nearby stimulus emerges from the local computation.

3.3 Natural Images Processing

We have also tested the model using natural images taken from a color CCD camera. No image processing has been performed on the image but a conversion to a gray-level representation. Figure 6 exhibits an example where a subpart of a computer keyboard has been shot. This allows to illustrate the main feature of the proposed model. If one look closely at the half retina representing the keyboard (upper left part of the figure), one can see that several letters (O, P, L, M) are eligible for attention focus and for ocular saccade. However, the retinotopic projection onto the model of the V1 area reduces quite naturally this set to letters O and L. The model of the SC is thus confronted with a choice between these two locations and the dynamic field theory, as it has been introduced in the previous section, ensures that only one location remains after competition. However it is hard to specify the exact conditions that make the model focus on the O instead of the L letter in the given example and the spatially compact shape of the O is certainly to be taken into account. This example also underlies the inherent difficulty in temporally organizing ocular saccades without any top-down control. If we were to let the model only react to its sensory input, it would certainly focus on the

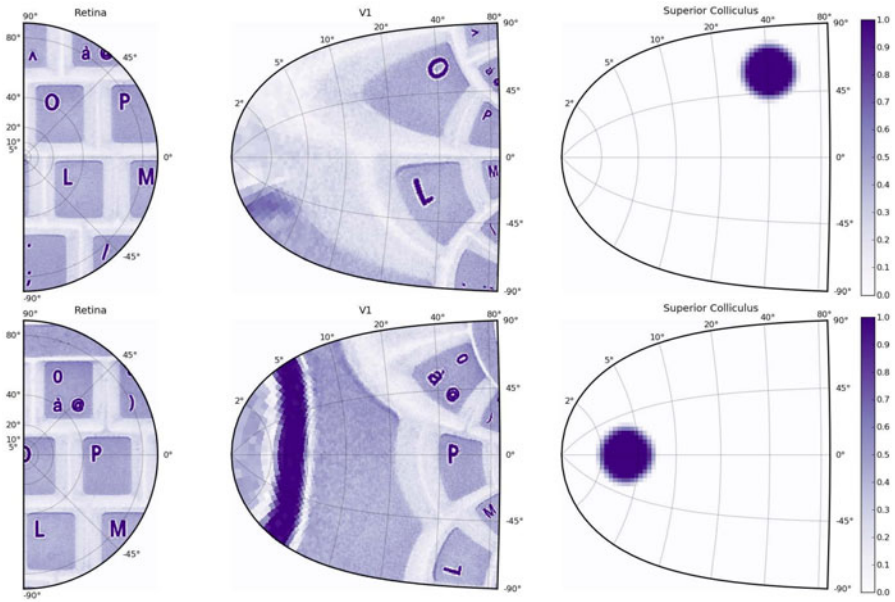


Fig. 6. An image of a computer keyboard has been captured using a color camera (resolution 1024×1280) and transformed into a normalized gray level image. **Upper figure.** The right half of the image is presented to the half retina area which in turn feeds the V1 area where retinotopy is applied following equation 2. The colliculus area enters a competition stage where most salient locations are eligible for final activation and after some iterations, the competition ends up onto the O letter that is thus considered the most salient location of the visual scene according to its location and activation. **Lower figure.** A saccade has been simulated to center the O letter into the center of the fovea and the colliculus now focuses onto a subpart of the O letter that appears to be the newly most salient location of the new visual scene.

most salient location without ever exploring other points of interest (from a behavioral point of view). If the actual saccade brings into view another salient location, the model would jump again to the new location (provided we inhibited the foveal region to prevent the model to be stuck forever on this single location) but in such a case, nothing would prevent the model from going to location A then location B and then again location A, being trapped in a cycle. Exploring the whole visual scene thus requires some kind of top down control to be able to dynamically inhibit visited location once they have been focused in order to favor other locations. This is out of scope of the present article but this has been already made in a wider but less realistic model [33].

4 Discussion

We have introduced in this paper a model of the superior colliculus based on on a large set of biological data. This model has been designed using a strongly constrained modeling framework relying on a set of four computational principles (distributed, asynchronous, numerical and adaptive) and those properties ensure to some extent that the

model does not suffer from usual artifacts of such computational framework (presence of a central supervisor deciding of the actual behavior). More specifically, the saccadic behavior we exhibited through the various experiments is a true and emergent property of local and homogeneous computations only. If we give a closer look to the selection process that is carried out when the model is presented with two identical stimuli (but at two different locations), we may explain the selection of the stimuli closest to the foveal region because of the cortical magnification. Said differently, the cortical magnification deeply influences the network topology and consequently the saliency of any presented stimuli. If we were to use some different magnification function, this would modify as well the selection process. This selective behavior is thus tightly linked to the spatial and physic implementation of the computational units. The intelligence of the system is thus rooted in its physical instantiation (even though it is simulated in our case).

However, if we now give a closer look to figure 3 we may realize that there is counterpart for such an automatic selection. Because the foveal region benefits from a much higher resolution than peripheral regions, the projections from retina to the V1 region distorts the geometrical properties of the image. This is especially the case of straight lines that are now projected as curved lines within the V1 region. Furthermore, the projection of any straight line from retina to V1 is unique and does not benefit from the same geometrical properties. How do we recognize a straight line in such conditions ? Classical answers relying on generic neighborhood functions that would (for example) link geometrically *aligned* neurons (hence mimicking the abstract description of a geometrical line) is not possible anymore. We thus have to change paradigm and consider new approaches like the one described in [34]. In this article, authors propose to reconsider vision by integrating the sensory-motor dimension of perception. Even though a straight line is not projected as a straight line in visual regions, there is nonetheless a physical property that remains true independently of the physical apparatus: if we move eyes along a straight line, there is an invariance in the projection because this is the physical definition of a line. Such sensori-motor behavior is a complex challenge that we are now actively exploring in terms of the temporal organization of the saccadic behavior. In order to achieve such active behavior, we now have to consider endogenous inputs conveying such information as instructions, goals or motivations from other higher-level neural structures. This lead us to consider anatomical structure such as the basal ganglia that are known to be deeply involved with voluntary motor control. In the end, we expect the model to achieve vision and recognition based on motor learning that may ultimately replace *passive perception*.

References

1. Ungerleider, L., Mishkin, M.: Two Cortical Visual Systems. In: Ingle, D.J., Goodale, M., Mansfield, R.J.W. (eds.) *Analysis of Visual Behaviour*, pp. 549–586 (1982)
2. Milner, A., Goodale, M.: *The Visual Brain in Action*. Oxford University Press (1995)
3. Goodale, M., Humphrey, G.: The objects of action and perception. *Cognition* 67, 181–207 (1998)
4. Ballard, D., Hayhoe, M., Pook, P., Rao, R.: Deictic Codes for the Embodiment of Cognition. *Behavioral and Brain Sciences* 20, 723–767 (1997)

5. Rizzolatti, G., Riggio, L., Dascola, I., Umil, C.: Reorienting attention across the horizontal and vertical meridians: evidence in favor of a premotor theory of attention. *Neuropsychologia* 25, 31–40 (1987)
6. Lettvin, J., Maturana, H., McCulloch, W., Pitts, W.: What the frog's eye tells the frog's brain. In: Corning, W.C., Balaban, M. (eds.) *The Mind: Biological Approaches to its Functions*, pp. 233–258 (1968)
7. Isa, T.: Intrinsic processing in the mammalian superior colliculus. *Current Opinion in Neurobiology* 12, 668–677 (2002)
8. Girard, B., Berthoz, A.: From brainstem to cortex: computational models of saccade generation circuitry. *Progress in Neurobiology* 77, 215–251 (2005)
9. Muller, J., Philiastides, M., Newsome, W.: Microstimulation of the superior colliculus focuses attention without moving the eyes. *Proceedings of the National Academy of Sciences* 102, 524–529 (2005)
10. Findlay, J., Walker, R.: A model of saccade generation based on parallel processing and competitive inhibition. *Behavioral and Brain Sciences* 22, 661–674 (1999)
11. Kramer, A.F., Irwin, D.E., Theeuwes, J., Hahn, S.: Oculomotor capture by abrupt onsets reveals concurrent programming of voluntary and involuntary saccades. *Behavioral and Brain Sciences* 22, 689–690 (1999)
12. Godijn, R., Theeuwes, J.: Programming of endogenous and exogenous saccades: evidence for a competitive integration model. *Journal of Experimental Psychology: Human Perception and Performance* 28, 1039–1054 (2002)
13. Amari, S.I.: Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics* 27, 77–87 (1977)
14. Trappenberg, T., Dorris, M., Munoz, D., Klein, R.: A model of saccade initiation based on the competitive integration of exogenous and endogenous signals in the superior colliculus. *Journal of Cognitive Neuroscience* 13, 256–271 (2001)
15. Schneider, S., Erhagen, W.: A neural field model for saccade planning in the superior colliculus: speed-accuracy tradeoff in the double-target paradigm. *Neurocomputing* 44–46, 623–628 (2002)
16. Wurtz, R., Optican, L.: Superior colliculus cell types and models of saccade generation. *Current Opinion in Neurobiology* 4, 857–861 (1994)
17. Marilly, E., Mercier, A., Coroyer, C., Faure, A., Cachard, O.: Propriétés d'un pré-processeur de vision fovéale. Dix-septième colloque GRETSI (1999)
18. Purves, D.: *Neurosciences*, 2nd edn. De Boeck (2004)
19. Robinson, D.: Eye movements evoked by collicular stimulation in the alert monkey. *Vision Research* 12, 1795–1808 (1972)
20. Ottes, F., Gisbergen, J.V., Eggermont, J.: Visuomotor fields of the superior colliculus: a quantitative model. *Vision Res.* 26, 857–873 (1986)
21. Bear, M., Connors, B., Paradiso, M.: *Neuroscience: Exploring the Brain*. Lippincott Williams & Wilkins (1996)
22. Wilson, H., Cowan, J.: A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Biological Cybernetics* 13, 55–80 (1973)
23. Taylor, J.: Neural bubble dynamics in two dimensions: foundations. *Biological Cybernetics* 80, 393–409 (1999)
24. Rougier, N., Vitay, J.: Emergence of attention within a neural population. *Neural Networks* 19, 573–581 (2006)
25. Munoz, D., Istvan, P.: Lateral inhibitory interactions in the intermediate layers of the monkey superior colliculus. *J. Neurophysiol.* 79, 1193–1209 (1998)
26. Sparks, D., Lee, C., Rohrer, W.: Population coding of the direction, amplitude, and velocity of saccadic eye movements by neurons in the superior colliculus. *Cold Spring Harbor Symposia on Quantitative Biology* 55, 805–811 (1990)

27. McIlwain, J.: Large receptive fields and spatial transformations in the visual system. *Int. Rev. Physiol.* 10, 223–248 (1976)
28. Sparks, D., Holland, R., Guthrie, B.: Size and distribution of movement fields in the monkey superior colliculus. *Brain Res.* 113, 21–34 (1976)
29. Lee, C., Rohrer, W., Sparks, D.: Population coding of saccadic eye movements by neurons in the superior colliculus. *Nature* 332, 357–360 (1988)
30. Yarbus, A.: *Eye movements and vision*, 1st edn. Plenum Press (1967)
31. Noton, D., Stark, L.: Scanpaths in eye movements during pattern perception. *Science* 13, 149–177 (1971)
32. Levy-Schoen, A.: Le champ d'activité du regard: données expérimentales. *Année Psychol.* 74, 43–66 (1974)
33. Fix, J., Vitay, J., Rougier, N.: A computational model of spatial memory anticipation during visual search. In: *Anticipatory Behavior in Adaptive Learning Systems* (2006)
34. O'Regan, J., Noe, A.: A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences* 24, 939–1031 (2001)

Use of Swarm Intelligence for the Identification of a Class of Nonlinear Dynamical Systems

Syed Z. Rizvi and Hussain N. Al-Duwaish

Department of Electrical Engineering
King Fahd University of Petroleum & Minerals, Saudi Arabia
{srizvi, hduwaish}@kfupm.edu.sa
<http://www.kfupm.edu.sa>

Abstract. In this work, properties of swarm intelligence are exploited to solve the identification problem of a class of nonlinear dynamical systems known as Hammerstein systems. Without any assumption on the structure of nonlinearity of the system, the nonlinearity is modeled using artificial neural network. Synaptic weights of the neural network are estimated via a particle swarm-based learning routine. Linear dynamics of the system are modeled using state space model. A recursive algorithm is developed to estimate the two components of the Hammerstein system. Numerical Monte-Carlo simulations are performed to test the reliability and repeatability of the identification technique presented. Identification is carried out with noisy data, having low signal-to-noise ratio (SNR). The presented identification technique provides encouraging estimation results.

Keywords: Particle swarm optimization, Linear dynamics, Neural network training, Nonlinearity, Artificial neural networks, Synaptic weights, Signal-to-Noise ratio.

1 Introduction

For the ease of estimation and study, nonlinear dynamical systems are often broken down into cascaded subsystems. A variety of such systems are collectively known as the block-oriented models. The Hammerstein system is one such block-oriented model. It is made up of a static nonlinear subsystem followed by a dynamical linear subsystem. Figure 1 shows the block diagram of a Hammerstein model. Variable $u(t)$ is the system input at time index t , $y(t)$ is the system's output, while $v(t)$ is an intermediate variable which is inaccessible to measurement. The challenge of Hammerstein model identification therefore lies in estimating the two subsystems based on measurement of input and output data only. The Hammerstein model has been used in the literature to model real-life nonlinear dynamical industrial processes, including a pH neutralization process [1], a heat exchanger [2], nonlinear filters [3], and a water heater [4] among several other processes. Hammerstein model identification is therefore of paramount importance to researchers, and as can be expected, it has attracted a lot of attention in the past few decades. Hammerstein system identification primarily falls under two main categories, parametric and nonparametric identification. Nonparametric models represent the system in terms of curves resulting from expansion of series such as the Volterra series or kernel regression. Parametric representations are more compact having fewer parameters. Notable parametric identification techniques can be found in [5],

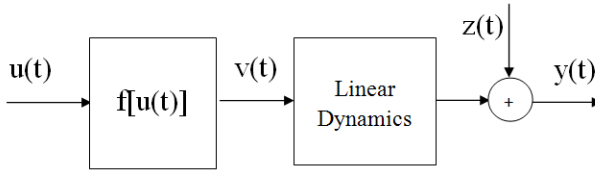


Fig. 1. Block diagram of a Hammerstein model

[6], [7], [8], [9] and in references therein. Nonparametric identification techniques can be found in several papers including, but not limited to those of [10], [11], [12].

Recently, advances in system identification have led to the development of more efficient and computationally less complicated identification techniques like subspace identification methods (SIM). However, its main limitation lies in that its use is restricted to linear systems only. Its advantages include being computationally less complicated as compared to conventional prediction error methods (PEM), not requiring initial estimate of a canonical model like PEM, and being easily extendable to systems having multiple inputs and outputs [13]. Owing to these advantages, it is not surprising that attempts have been made to extend the use of SIM to nonlinear systems such as Wiener and Hammerstein systems. Such works include the use of static nonlinearity in the feedback path [14], Hammerstein systems with known nonlinearity structures [15], and using least squares support vector machines [16].

On the other hand, biologically inspired estimation and optimization algorithms have changed the way we approach problems to a significant degree. With the publication of Kennedy and Eberhart's paper on particle swarm optimization (PSO) [17], a door opened for new optimization techniques based on human and animal social behavior. These algorithms have proven extremely efficient in solving problems based on high-dimensional, multi-modal, discontinuous as well as continuous functions.

Here, we present a new estimation method for Hammerstein system identification, which exploits the properties of both subspace methods for estimation, as well as those of swarm intelligence, to obtain accurate nonlinear models. The method presented uses radial basis function (RBF) neural network in cascade with a state space model. A recursive algorithm is presented for parameter estimation of the two subsystems using the above mentioned techniques. Notation convention used in this work is as follows. Lower case variables represent scalars. Lower case bold variables represent vectors. Upper case bold letters denote matrices. The only exception to this convention lies in the choice of a more conventional J for the cost function.

2 Identification Scheme

In this section, we take a look at the proposed scheme for the identification of Hammerstein model. Model structure in this work uses state space model to represent the linear dynamic part. The memoryless nonlinear part is modeled using an RBF network. An RBF network is an effective type of neural network that has proved useful in applications like function approximation and pattern recognition. A typical three-layer RBF network is shown in Figure 2. The input layer connects the network to its environment.

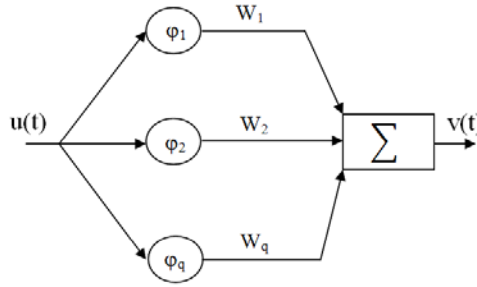


Fig. 2. An RBF neural network with q neurons in the hidden layer

The second layer, known as the hidden layer, performs a fixed nonlinear transformation using basis functions. The output layer linearly weighs the response of the network to the output [18]. The external inputs to the system $u(t)$ are fed to the RBF network, which generates the outputs $v(t)$. Considering an RBF network having q number of neurons in the hidden layer, the basis vector is

$$\phi(t) = [\phi\|u(t) - c_1\| \cdots \phi\|u(t) - c_q\|]^T, \quad (1)$$

where c_i is the chosen center for the i^{th} neuron, $\|\cdot\|$ denotes norm that is usually Euclidean, and ϕ_i is the nonlinear radial basis function for the i^{th} neuron, given by

$$\phi_i(t) = \exp\left(-\frac{\|u(t) - c_i\|^2}{2\sigma^2}\right), \quad (2)$$

where σ is the spread of the Gaussian function $\phi_i(t)$. If the set of output layer weights of the RBF network is given by

$$\mathbf{w} = [w_1 \quad w_2 \cdots w_q]^T, \quad (3)$$

the RBF output $v(t)$ is given by

$$v(t) = \mathbf{w}^T \phi(t). \quad (4)$$

Considering a system with a single input and output, the output of the RBF network $v(t)$ in turn acts as input to the state space model translating it into final output $y(t)$. The equation for $y(t)$ is given by discrete-time state space equation

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}v(t) + \tilde{\mathbf{w}}(t), \quad (5)$$

$$y(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}v(t) + z(t), \quad (6)$$

where $v(t)$ and $y(t)$ are input and output of the state space system at discrete-time index t , $z(t)$ and $\tilde{\mathbf{w}}(t)$ are the measurement and process noise.

The problem of Hammerstein modeling is therefore formulated as follows. Given a set of m measurements of noisy inputs $u(t)$ and outputs $y(t)$, the problem is reduced to finding the weights of the RBF network and the matrices of the state space model.

For the estimation of state space matrices, N4SID numerical algorithm for subspace identification [19] is used. The algorithm determines the order n of the system, the system matrices $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times p}$, $\mathbf{C} \in \mathbb{R}^{r \times n}$, $\mathbf{D} \in \mathbb{R}^{r \times p}$, covariance matrices $\mathbf{Q} \in \mathbb{R}^{n \times n}$, $\mathbf{R} \in \mathbb{R}^{1 \times 1}$, $\mathbf{S} \in \mathbb{R}^{n \times 1}$, and the Kalman gain matrix \mathbf{K} , where p denotes the number of inputs and r denotes the number of outputs of the system, without any prior knowledge of the structure of the system, given that a large number of measurements of inputs and outputs generated by the unknown system of equations (5) and (6) is provided. In N4SID, Kalman filter states are first estimated directly from input and output data, then the system matrices are obtained [19].

For Hammerstein identification problem, it is desired that the error between the output of the actual system, and that of the estimated model be minimized. Therefore, in a way this becomes an optimization problem where a cost index is to be minimized. For the system described in equations (1)-(6), the cost index is given by

$$J = \sum_{t=1}^m e^2(t) = \sum_{t=1}^m (y(t) - \hat{y}(t))^2, \tag{7}$$

where $y(t)$ and $\hat{y}(t)$ are the outputs of the actual and estimated systems at time index t . The weights of the RBF network are therefore updated so as to minimize this cost index. For this purpose, PSO used.

PSO is a heuristic optimization algorithm which works on the principle of swarm intelligence [21]. It imitates animals living in a swarm collaboratively working to find their food or habitat. In PSO, the search is directed, as every particle position is updated in the direction of the optimal solution. It is robust and fast and can solve most complex and nonlinear problems. It generates globally optimum solutions and exhibits stable convergence characteristics. In this work, PSO is used to train the RBF network. Each particle of the swarm represents a candidate value for the weight of the output layer of RBF network. The fitness of the particles is the reciprocal of the cost index given in equation (7). Hence, the smaller the sum of output errors, the more fit are the particles. Based on this principle, PSO updates the position of all the particles moving towards an optimal solution for the weights of RBF neural network.

The i^{th} particle of the swarm is given by a k -dimension vector $\tilde{\mathbf{x}}_i = [\tilde{x}_{i1} \cdots \tilde{x}_{ik}]$, where k denotes the number of optimized parameters. Similar vectors $\tilde{\mathbf{p}}_i$ and $\tilde{\mathbf{v}}_i$ denote the best position and velocity of the i^{th} particle respectively. The velocity of the i^{th} particle is updated as

$$\begin{aligned} \tilde{\mathbf{v}}_i(t + 1) = & \chi[w\tilde{\mathbf{v}}_i(t) + c_1r_1(t)\{\tilde{\mathbf{p}}_i(t) - \tilde{\mathbf{x}}_i(t)\} \\ & + c_2r_2(t)\{\tilde{\mathbf{p}}_g(t) - \tilde{\mathbf{x}}_i(t)\}], \end{aligned} \tag{8}$$

and the particle position is updated as

$$\tilde{\mathbf{x}}_i(t + 1) = \tilde{\mathbf{x}}_i(t) + \tilde{\mathbf{v}}_i(t + 1). \tag{9}$$

In the above equations, $\tilde{\mathbf{p}}_g$ denotes global best positions, while c_1 and c_2 are the *cognitive* and *social* parameters respectively, and are both positive constants. Parameter w is the *inertia weight* and χ is called the *constriction factor* [22]. The value of cognitive

parameter c_1 signifies a particle’s attraction to a local best position based on its past experiences. The value of social parameter c_2 determines the swarm’s attraction towards a global best position.

3 Learning Algorithm

Given a set of m observations of input and output, $\mathbf{u} \in \mathcal{R}^{1 \times m}$ and $\mathbf{y} \in \mathcal{R}^{1 \times m}$, a hybrid PSO/Subspace identification algorithm is proposed below.

1. Estimate state space matrices $\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0$ and \mathbf{D}_0 (initial estimate) from original non linear data using N4SID.
2. Iteration = $k = 1$.
3. Initialize RBF network with neuron centers evenly distributed in the input space.
4. Initialize PSO with random population of possible RBF network weights.
5. $\mathbf{w}_k = \operatorname{argmin}_{\mathbf{w} \in \mathcal{R}^q} J(\mathbf{A}_{k-1}, \mathbf{B}_{k-1}, \mathbf{C}_{k-1}, \mathbf{D}_{k-1}, \mathbf{w})$.
6. With optimum weights obtained above, estimate set of RBF neural network outputs $\mathbf{v}_k \in \mathcal{R}^{1 \times m}$

$$\begin{aligned} \mathbf{v}_k &= \mathbf{w}_k^T \Phi \\ &= [w_{1k} \cdots w_{qk}] \begin{bmatrix} \phi_1(1) \cdots \phi_1(m) \\ \vdots \\ \phi_q(1) \cdots \phi_q(m) \end{bmatrix}. \end{aligned}$$

7. Estimate state space matrices $\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k$ and \mathbf{D}_k from $[\mathbf{v}_k \ \mathbf{y}]^T$. This estimate of state space model would be an improvement on the previous estimate.
8. Regenerate $\hat{\mathbf{y}}_k \in \mathcal{R}^{1 \times m}$.
9. If minimum goal is not achieved, $k = k + 1$. Repeat steps 4 to 7.

4 Simulation Results

4.1 Example 1

The first example considers the following Hammerstein type nonlinear process whose static nonlinearity is given by

$$v(t) = \operatorname{sign}(u(t)) \sqrt{|u(t)|}. \tag{10}$$

The dynamic linear part is given by a third order discrete-time state space system

$$\mathbf{A} = \begin{bmatrix} 1.80 & 1 & 0 \\ -1.07 & 0 & 1 \\ 0.21 & 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 4.80 \\ 1.93 \\ 1.21 \end{bmatrix},$$

$$\mathbf{C} = [1 \ 0 \ 0].$$

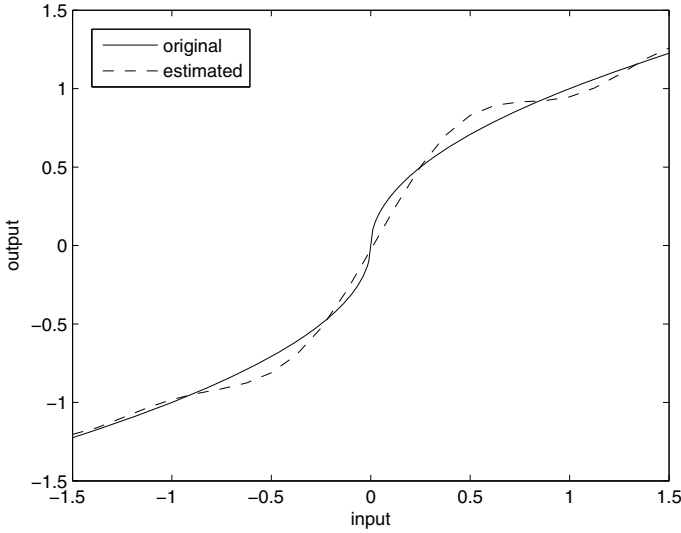


Fig. 3. Estimate of square root nonlinearity of example 1

The eigenvalues of the linear subsystem lie at $\lambda_1 = 0.7$, $\lambda_2 = 0.6$, and $\lambda_3 = 0.5$. Desired outputs are generated by exciting the process model with a rich set of uniformly distributed random inputs in the interval $[-1.75, 1.75]$. An RBF network of 25 neurons is initialized with random synaptic weights, and centers uniformly distributed in the input interval. The choice of the number of neurons is non-trivial. A suitable number of neurons can be chosen while designing based on experience. Repeated simulations can then be run to test the performance increasing the number of neurons each time until no appreciable performance improvement is noticed. PSO social and cognitive parameters c_1 and c_2 are kept almost equal to each other with c_1 slightly larger than c_2 and $c_1 + c_2 \geq 4$ as proposed in [23]. This allows trusting past experiences slightly more than those of one's neighbors. Constriction factor is kept close to 1 to enable slow convergence with better exploration. Number of particles amount to 25 for the synaptic weights of 25 neurons. A swarm population size of 50 is selected and the optimization process is run for 100 iterations.

The algorithm shows promising results and mean squared output error between normalized outputs of actual and estimated systems converges to a final value of 4×10^{-4} in 10 iterations of the algorithm. Figure 3 shows the nonlinearity estimate. An easy way to evaluate the estimate of linear dynamic part lies in comparing the eigenvalues of the estimated subsystem with true ones. The eigenvalues of the estimated subsystem lie at $\hat{\lambda}_1 = 0.72$, $\hat{\lambda}_2 = 0.53$ and $\hat{\lambda}_3 = 0.53$. Figure 4 shows the step response of the dynamic linear part.

To evaluate the performance of the proposed algorithm in noisy environment, zero mean Gaussian additive noise is included at the output of the system such that the signal-to-noise ratio (SNR) is 10dB. The algorithm performs well in estimating the system despite low output SNR. The final mean squared error between normalized outputs converges to 1.4×10^{-3} in 13 iterations. Nonlinearity estimate is shown in

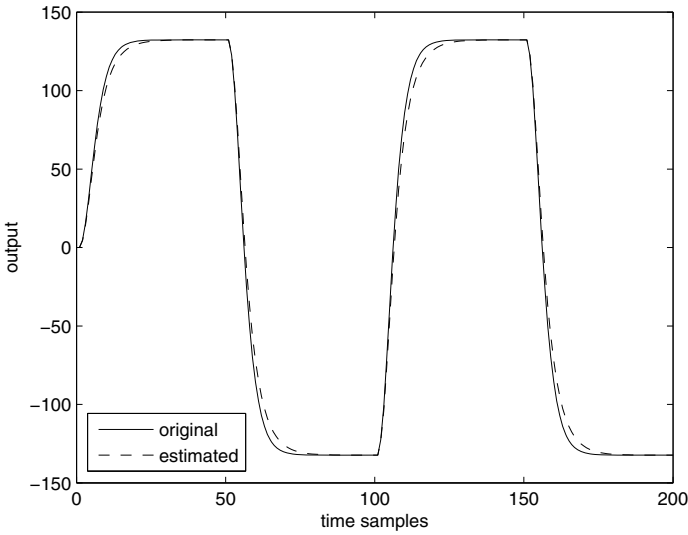


Fig. 4. Step response of linear dynamic subsystem of example 1

Figure 5. The eigenvalues of the estimated system lie at $\hat{\lambda}_1^{10dB} = 0.73$, $\hat{\lambda}_2^{10dB} = 0.73$ and $\hat{\lambda}_1^{10dB} = 0.58$.

The results presented above are obtained from a single run of estimation algorithm. To further ensure the reliability and repeatability of the algorithm, Monte-Carlo simulation is carried out and ensemble statistics are tabulated in Table 1. The statistics show encouraging convergence of normalized output squared error and estimation of linear subsystem. Parameters of the nonlinearity cannot be compared because of the nonparametric nature of estimation. At best, the estimates of nonlinearity can be judged from the shapes of estimated nonlinear function as presented in Figures 3 and 5.

4.2 Example 2

The second example considers the following Hammerstein type nonlinear process whose static nonlinearity is given by

$$v(t) = \tanh [2u(t)] \quad 1.5 \geq u(t),$$

$$v(t) = \frac{\exp(u(t)) - 1}{\exp(u(t)) + 1} \quad 4 > u(t) > 1.5.$$

The dynamic linear part is given by the following second order discrete-time state space system

$$\mathbf{A} = \begin{bmatrix} 1.0 & 1.0 \\ -0.5 & 0.0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix},$$

$$\mathbf{C} = [1 \quad 0].$$

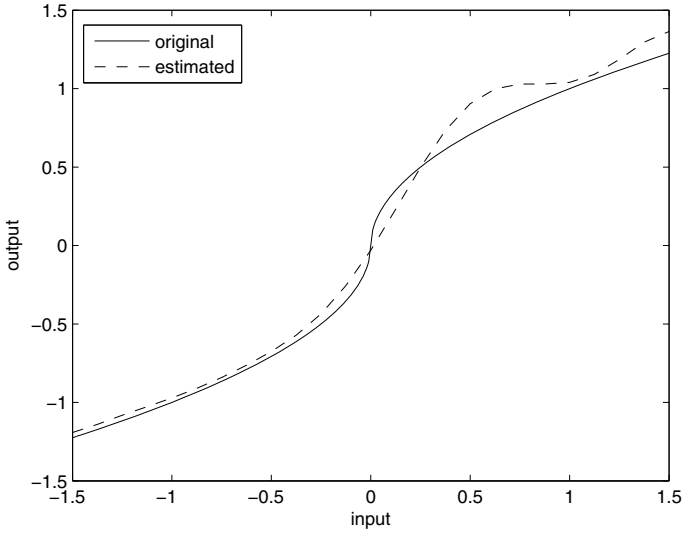


Fig. 5. Estimate of square root nonlinearity of example 1 with output SNR 10dB

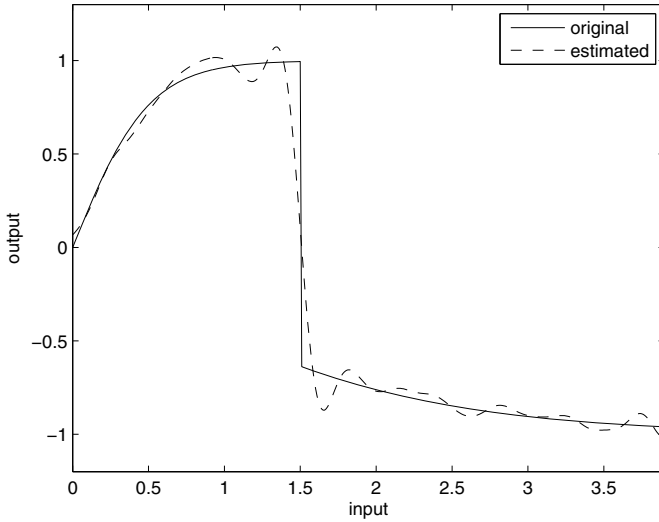


Fig. 6. Estimate of tangent-hyperbolic nonlinearity of example 2

The linear part of the system has eigenvalues at $\lambda_{1,2} = 0.5 \pm 0.5i$. Desired outputs are generated by exciting the process model with a rich set of uniformly distributed random input in the interval $[0, 4]$. An RBF network with 25 neurons is initialized with random synaptic weights and uniformly distributed centers chosen within the input interval. PSO social and cognitive parameters and constriction factor are kept similar to example 1. The number of particles is equal to the number of neurons and a population size of 50 gives good results again. The algorithm performs well and estimates the system in

Table 1. Monte-Carlo simulation statistics for example 1

Estimation results without output noise	
Total number of runs	200
Magnitude of actual eigenvalue λ_1 of linear subsystem	0.7
Mean magnitude of estimated eigenvalue $\hat{\lambda}_1$ of linear subsystem	0.713
Variance of eigenvalue estimate $\hat{\lambda}_1$	8×10^{-4}
Magnitude of actual eigenvalue λ_2 of linear subsystem	0.6
Mean magnitude of estimated eigenvalue $\hat{\lambda}_2$ of linear subsystem	0.62
Variance of eigenvalue estimate $\hat{\lambda}_2$	4.5×10^{-3}
Magnitude of actual eigenvalue λ_3 of linear subsystem	0.5
Mean magnitude of estimated eigenvalue $\hat{\lambda}_3$ of linear subsystem	0.481
Variance of eigenvalue estimate $\hat{\lambda}_3$	5.7×10^{-3}
Average number of iterations required for every run	8.26
Average mean squared output error (MSE) normalized over [0,1]	9.8×10^{-4}
Estimation results with output SNR 10dB	
Total number of runs	200
Magnitude of actual eigenvalue λ_1 of linear subsystem	0.7
Mean magnitude of estimated eigenvalue $\hat{\lambda}_1$ of linear subsystem	0.75
Variance of eigenvalue estimate $\hat{\lambda}_1$	1.6×10^{-3}
Magnitude of actual eigenvalue λ_2 of linear subsystem	0.6
Mean magnitude of estimated eigenvalue $\hat{\lambda}_2$ of linear subsystem	0.68
Variance of eigenvalue estimate $\hat{\lambda}_2$	6×10^{-3}
Magnitude of actual eigenvalue λ_3 of linear subsystem	0.5
Mean magnitude of estimated eigenvalue $\hat{\lambda}_3$ of linear subsystem	0.4
Variance of eigenvalue estimate $\hat{\lambda}_3$	6×10^{-2}
Average number of iterations required for every run	8.02
Average mean squared output error (MSE) normalized over [0,1]	6.6×10^{-3}

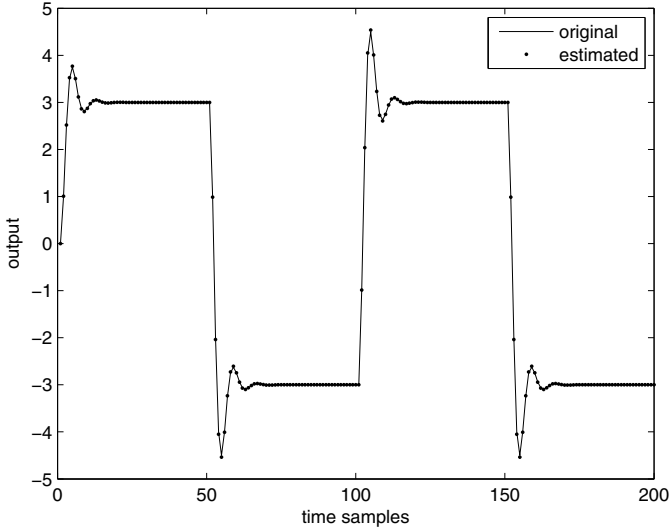


Fig. 7. Step response of linear dynamic subsystem of example 2

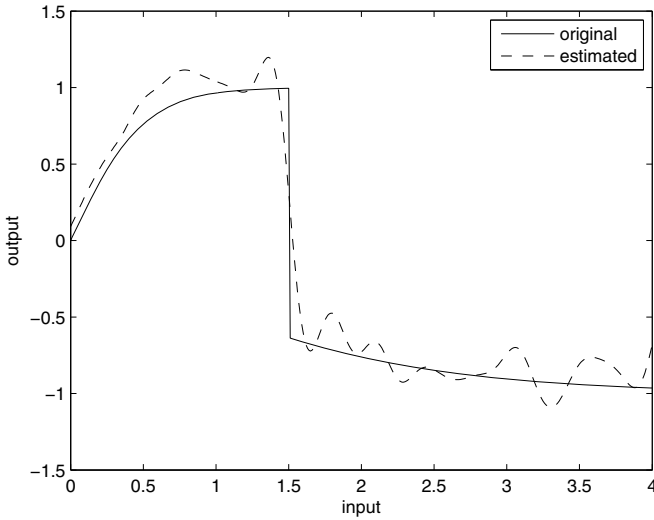


Fig. 8. Estimate of tangent-hyperbolic nonlinearity of example 2 with output SNR 10dB.

12 iterations. The mean squared output error between normalized values of actual and estimated outputs converges to a final value of 8×10^{-4} . The estimate of nonlinearity shape is shown in Figure 6. Eigenvalues of the estimated system lie at $\hat{\lambda}_{1,2} = 0.497 \pm 0.5i$. Step response of linear subsystem is shown in Figure 7.

Estimation is then carried out in noisy environment, with zero mean Gaussian additive noise included at the output of the system such that the signal-to-noise ratio (SNR)

is 10dB. The algorithm performs well in noisy environment. The final mean squared error converges to 1.8×10^{-3} in 13 iterations. Nonlinearity estimate is shown in Figure 8. The eigenvalues of the estimated system lie at $\hat{\lambda}_{1,2}^{10dB} = 0.493 \pm 0.499i$.

Table 2 shows ensemble statistics of Monte-Carlo simulation for example 2. The statistics show encouraging convergence of normalized output squared error and estimation of linear subsystem. As mentioned before, parameters of the nonlinearity cannot be compared due to the nonparametric nature of estimation. At best, the estimates of nonlinearity can be judged from the shapes of estimated nonlinear function as presented in Figures 6 and 8.

Table 2. Monte-Carlo simulation statistics for example 2

Estimation results without output noise	
Total number of runs	200
Magnitude of actual eigenvalues $\lambda_{1,2}$ of linear subsystem	0.7071
Mean magnitude of estimated eigenvalues $\hat{\lambda}_{1,2}$ of linear subsystem	0.7071
Variance of eigenvalue estimates	2×10^{-5}
Average number of iterations required for every run	9
Average mean squared output error (MSE) normalized over [0,1]	7×10^{-4}
Estimation results with output SNR 10dB	
Total number of runs	200
Magnitude of actual eigenvalues $\lambda_{1,2}$ of linear subsystem	0.7071
Mean magnitude of estimated eigenvalues $\hat{\lambda}_{1,2}$ of linear subsystem	0.7079
Variance of eigenvalue estimates	6×10^{-5}
Average number of iterations required for every run	21
Average mean squared output error (MSE) normalized over [0,1]	2×10^{-3}

5 Further Enhancement to the Learning Algorithm

Seeking further improvement, we propose searching for optimum basis function centers, as opposed to initializing them with an even distribution in the input space and then keeping them fixed throughout the neural network learning. This way, particle swarm would be used to find the optimum combination of synaptic weights and basis function centers. A slight modification is therefore made to the learning algorithm of Section 3. For a similar set of m observations of input and output, $\mathbf{u} \in \mathbb{R}^{1 \times m}$ and $\mathbf{y} \in \mathbb{R}^{1 \times m}$, the modified learning algorithm would stand similar to the previous algorithm of Section 3, except that in the modified algorithm, the number of particles in the swarm would amount to the sum of the number of synaptic weights and neuron centers. Step 4 of the algorithm would change to

$$[\mathbf{w}_k, \mathbf{c}_k] = \underset{\mathbf{w} \in \mathbb{R}^q, \mathbf{c} \in \mathbb{R}^q}{\text{argmin}} J(\mathbf{A}_{k-1}, \mathbf{B}_{k-1}, \mathbf{C}_{k-1}, \mathbf{D}_{k-1}, \mathbf{w}, \mathbf{c}).$$

Table 3. Monte-Carlo simulation statistics for example 1. Algorithm 1 represents algorithm of Section 3 Algorithm 2 represents enhanced learning algorithm of Section 5

Estimation results	Alg. 1	Alg. 2
Total number of runs	200	200
Magnitude of actual eigenvalue λ_1 of linear subsystem	0.7	0.7
Mean magnitude of estimated eigenvalue $\hat{\lambda}_1$ of linear subsystem	0.713	0.7031
Variance of eigenvalue estimate $\hat{\lambda}_1$	8×10^{-4}	7.9×10^{-4}
Magnitude of actual eigenvalue λ_2 of linear subsystem	0.6	0.6
Mean magnitude of estimated eigenvalue $\hat{\lambda}_2$ of linear subsystem	0.62	0.6136
Variance of eigenvalue estimate λ_2	4.5×10^{-3}	3.7×10^{-3}
Magnitude of actual eigenvalue λ_3 of linear subsystem	0.5	0.5
Mean magnitude of estimated eigenvalue λ_3 of linear subsystem	0.481	0.4934
Variance of eigenvalue estimate $\hat{\lambda}_3$	5.7×10^{-3}	3.3×10^{-3}
Average number of iterations required for every run	8.26	8.64
Average mean squared output error (MSE) normalized over [0,1]	9.8×10^{-4}	2.9×10^{-4}

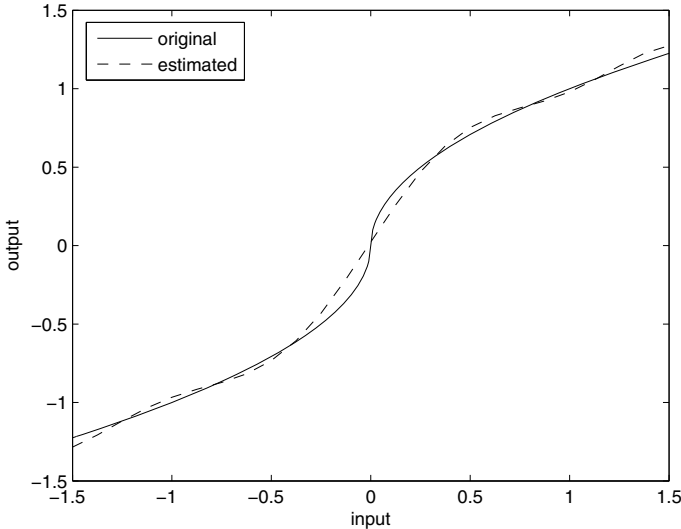


Fig. 9. Estimate of square root nonlinearity of example 1 with enhanced learning

It is observed that searching for optimum neuron centers along with the weights yields much improved identification results. Problem of Example 1 is tested again with enhanced learning algorithm. Figure 9 shows an improved nonlinearity estimate. Table 3 compares Monte-Carlo simulations statistics for both the algorithms. Estimates of all eigenvalues of the linear subsystem are closer to actual values in the enhanced

learning algorithm. Most noteworthy is the improvement in the overall average output mean squared error. Improved accuracy of identification however, comes at the price of computation complexity, since the enhanced learning algorithm involves learning of both the neuron centers and the weights. In order to reduce computation complexity, different variants of swarm intelligence-based algorithms present in the literature can be used.

6 Conclusions

A subspace and swarm intelligence-based identification algorithm is presented here. The combined algorithm reaps the benefits of both techniques. The convergence properties of the presented algorithm are directly related to the convergence properties of PSO and subspace algorithms. PSO has been usually known to perform better than most evolutionary algorithms (EA)s in finding global optimum provided its parameters are tuned properly according to the application. The subspace algorithm is also known for having no convergence problems. Its numerical robustness is guaranteed because of well understood linear algebra techniques like QR decomposition and singular value decomposition (SVD). As a consequence, it does not experience problems like lack of convergence, slow convergence or numerical instability [19]. To gauge the ability of the combined algorithm to converge substantially, and to assure its repeatability and reliability, Monte-Carlo simulations have been carried out. Statistics presented in Tables 1, 2 and 3 show strong convergence and consistent performance of the proposed algorithms. The effect of noise is also studied, and the algorithm is seen to converge sufficiently in the presence of noisy data as shown in the simulation results.

Acknowledgements. The authors would like to acknowledge the support of King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia. This work also benefited immensely from the valuable comments of ICNC 2010 conference referees and attendants.

References

1. Fruzzetti, K.P., Palazoglu, A., McDonald, K.A.: Nonlinear model predictive control using Hammerstein models. *J. Proc. Control* 7, 31–41 (1997)
2. Eskinat, E., Johnson, S.H., Luyben, W.L.: Use of Hammerstein models in identification of nonlinear systems. *AIChE Journal* 37, 255–268 (1991)
3. Haddad, A.H., Thomas, J.B.: On optimal and suboptimal nonlinear filters for discrete inputs. *IEEE Trans. Information Theory* 14, 16–21 (1968)
4. Abonyi, I., Nagy, L., Szeifert, E.: Hybrid fuzzy convolution modelling and identification of chemical process systems. *Int. J. Systems Science* 31, 457–466 (2000)
5. Narendra, K.S., Gallman, P.: An iterative method for the identification of nonlinear systems using Hammerstein model. *IEEE Trans. Automatic Control* 11, 546–550 (1966)
6. Billings, S.: Identification of nonlinear systems - A survey. *IEEE Proc.* 127, 272–285 (1980)
7. Al-Duwaish, H.: A Genetic Approach to the Identification of Linear Dynamical Systems with Static Nonlinearities. *Int. J. Systems Science* 31, 307–314 (2001)

8. Vörös, J.: Modeling and Parameter Identification of Systems with Multisegment Piecewise-Linear Characteristics. *IEEE Trans. Automatic Control* 47, 184–188 (2002)
9. Wenxiao, Z.: Identification for Hammerstein Systems Using Extended Least Squares Algorithm. In: 26th Chinese Control Conference, pp. 241–245. IEEE Press, China (2007)
10. Greblicki, W.: Non-parametric orthogonal series identification of Hammerstein systems. *Int. J. Systems Science* 20, 2335–2367 (1989)
11. Al-Duwaish, H., Nazmulkarim, M., Chandrasekar, V.: Hammerstein model identification by multilayer feedforward neural networks. *Int. J. Systems Science* 28, 49–54 (1997)
12. Zhao, W., Chen, H.: Recursive identification for Hammerstein systems with ARX subsystem. *IEEE Trans. Automatic Control* 51, 1966–1974 (2006)
13. Katayama, T.: Subspace Methods for System Identification. Springer, London (2005)
14. Luo, D., Leonessa, A.: Identification of MIMO Hammerstein Systems with Nonlinear Feedback. In: American Control Conference 2002, pp. 3666–3671. IEEE Press, Galesburg (2002)
15. Verhaegen, M., Westwick, D.: Identifying MIMO Hammerstein systems in the context of subspace model identification methods. *Int. J. Control* 63, 331–349 (1996)
16. Goethals, I., Pelckmans, K., Suykens, J.A.K., Moor, B.D.: Identification of MIMO Hammerstein Models using Least Squares Support Vector Machines. *Automatica* 41, 1263–1272 (2005)
17. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE Press (1995)
18. Haykin, S.: *Neural Networks - A Comprehensive Foundation*. Prentice-Hall, New Jersey (1999)
19. Overschee, P.V., Moor, B.D.: N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica* 30, 75–93 (1994)
20. Overschee, P.V., Moor, B.D.: *Subspace Identification for Linear Systems: Theory-Implementation-Applications*. Kluwer Academic Publishers, Dordrecht (1996)
21. Kennedy, J., Eberhart, R.: *Swarm Intelligence*. Academic Press (2001)
22. Eberhart, R., Shi, Y.: Parameter Selection in Particle Swarm Optimisation. In: *Evolutionary Programming VII*, pp. 591–600 (1998)
23. Carlisle, A., Dozier, G.: An Off-The-Shelf PSO. In: *The Particle Swarm Optimization Workshop*, pp. 1–6 (2001)

Practical Graph Isomorphism for Graphlet Data Mining in Protein Structures

Carsten Henneges¹, Christoph Behle², and Andreas Zell¹

¹ Center for Bioinformatics Tübingen, Eberhard Karls Universität Tübingen
Sand 1, Tübingen, Germany

² Lehrstuhl für Theoretische Informatik, Eberhard Karls Universität Tübingen
Sand 13, Tübingen, Germany
carsten.henneges@uni-tuebingen.de
behlec@informatik.uni-tuebingen.de

Abstract. The tertiary structure of proteins is composed of α -helices and β -sheets being referred to as Secondary Structure Elements (SSE). SSE are evolutionary conserved and define the overall fold of a protein. Therefore they can be used to classify protein families. SSE form pairwise energetical interactions which can be described by graphs. Neighbourhood graphs employ edge conditions to filter relevant interactions out of a set of pairwise relations. Graphlet analysis then employs stochastic sampling of subgraphs to identify overrepresented interaction patterns. To distinguish graphlets while sampling requires an efficient algorithm for graph isomorphism.

In this Chapter, we describe a graph isomorphism algorithm that is easy to implement. In a preprocessing phase, the presented algorithm combines marks assigned to vertices to more informative marks. Propagated over edges, marks collect information about the structure of the graph and hence allow to efficiently find isomorphisms in a subsequent backtracking step.

Applying graphlet analysis to neighbourhood graphs for structures from the ICGEB Protein Benchmark database and the Super-Secondary Structure database (SSSDB), we identify 627 significant graphlets. Subsequently trained decision trees on these features predict the four SCOP levels and SSSDB classes with a mean Area Under Curve (mAUC) better than 0.89 (5-fold CV). Regularizing these decision trees to avoid overfitting reveals that for reliable prediction of structural features about 20 graphlets are sufficient. Especially, we find that graphlets composed of five secondary structure elements are most informative for classification. Conversely, using decision trees trained on the mere sequence of SSE obtained from the protein sequence we are also able to predict graphlets directly from secondary structure annotation. Optimal prediction performance thereby reaches up to a Matthews Correlation Coefficient (MCC) of 0.7.

From our experiments in this Chapter, we conclude that SSE interactions form patterns significantly different from random. These patterns are both useful to predict structural protein features as well as they can be predicted from protein sequence. Therefore they can be used as constraints to facilitate the *de novo* prediction of unknown protein structures.

Keywords: Graphlets, Data Mining, Relative Neighbourhood Graph, Secondary Structure Elements, Decision Tree Model Selection.

1 Introduction

Secondary structure elements (SSEs), β -sheets and α -helices, are important building blocks of proteins and interactions between these SSEs stabilize protein tertiary structures. Many categorization schemes are based on these SSEs like the SCOP [1] or CATH [2] databases. For example, the SCOP categorization scheme has four levels, each one giving more detailed information about the SSEs of a protein domain and the arrangement of its SSEs.

An important question, especially in the context of SSE-based categorization schemes, is whether there are interaction patterns between these SSEs that occur preferably in native protein structures or in protein structures of a given fold or superfamily. As SSEs are complex three-dimensional structures, whose relative states are hard to encode, analysing SSE interactions in full atomic detail can become complex. However, encoding energetic interaction between SSEs as graphs makes this problem amenable for graph-based analysis of protein structures. Thereby the SSEs of a structure are mapped to vertices of a graph, and interactions between SSEs are represented by edges. Graph mining methods [3] then can be applied on these reduced representations of protein structures.

At the heart of each graph mining application lies the problem of graph isomorphism (GI). For any pair of graphs having equal number of nodes and edges, it consists of deciding whether the two graphs are structurally equal, i.e., if there exists an adjacency preserving mapping between both sets of nodes. This mapping is referred to as the isomorphism.

Computing isomorphism for two given graphs G and H is a combinatorial search task. In the worst case, it amounts to searching all $|V|!$ combinations of nodes $v \in V$. While the problem of subgraph isomorphism is known to be NP-complete, this question is still open for GI. Nonetheless there exist polynomial algorithms for special classes of graphs (e.g. planar graphs) as well as practical implementations for general graphs (e.g. `nauty`, `VF/VF2`), written in highly optimised C. However, both alternatives, the restriction to a class of graphs as well as the usage of the hard to integrate implementations of GI solvers, is often not practical when working with graphlets. Here, we complement our previous paper [4] by a description of a practical graph isomorphism method that applicable for any class of graphs, is easy to implement and reveals fast runtime behaviour.

Using this algorithm, we mine graph representations of protein structures for subgraphs, termed graphlets, that preferably occur in native protein structures. The significance of a graphlet is statistically tested using a randomised graphs, serving as a background model [5]. This background model allows us to identify graphlets that occur more often in protein graphs than in random graphs. The value of the extracted graphlets is then demonstrated by using them in predictors for SCOP levels and SSSDB motifs.

2 Materials and Methods

Databases

We perform our experiments using protein structures from the SCOP40mini dataset from the ICGB database and the database of super-secondary structures SSSDB [6,7].

Protein Preparation

We use C++ routines and the BALL library [8] to analyze protein structures. As a first step, hydrogen atoms are added. After a consistency check of each residue, we assign partial charges and atom radii to atoms according to the AMBER force field. Finally, all hetero residues are removed from the structure.

To be independent from protein annotation, all SSEs are recomputed using the DSSP algorithm [9], afterwards all residues within loops are removed. α -helices are not further modified, whereas all connected components of β -strands linked by hydrogen bonds are merged into β -sheets. In this way, each β -sheet is treated as a complete 3D structure during the computation of interaction energies.

Graphical Encoding of Energetic SSE Interactions

To prepare the graph construction, we compute the pairwise matrix of SSE interaction energies. Let A and B be any secondary structures in a protein, and let $E[A, B, \dots]$ be the AMBER energy of a set of SSEs, then the pairwise interaction energy $I[A, B]$ is given as

$$I[A, B] = E[A, B] - E[A] - E[B]. \quad (1)$$

Graph construction serves for structural normalization as well as extracting the interaction model. It filters needless relations, while being independent from the computed amount of $I[A, B]$ energy and the relative distance of the SSEs. Therefore, it makes proteins having different numbers of SSEs comparable.

One convenient graph for this task is the Relative Neighbourhood Graph (RNG) [10]. The RNG connects two labelled SSE nodes if the following edge condition

$$I[A, B] \leq \max_C \{I[A, C], I[B, C]\}, \quad (2)$$

holds, where A, B, C are SSEs from the protein and $A \neq B \neq C$. The RNG is a connected proximity graph and, therefore, also connects SSEs that are too distant for direct protein residue contacts. As its edge condition resembles an ultra-metric [11], the RNG has shown great robustness in practice and is a powerful tool to extract meaningful perceptual structures [10].

A Practical Graph Isomorphism Method

To prepare the subsequent graphlet analysis, we describe our variant of the Weisfeiler-Lehmann refinement method for graph isomorphism [12], which is easy to implement and offers practical runtime performance¹. It consists of two phases, a preprocessing phase that extracts connectivity information out of each graph and a backtracking search for the isomorphism. By precomputing marks encoding the connectivity of a graph first, the algorithm is able to prune unnecessary comparisons as well as search space when computing isomorphisms in the second phase. We will show that the precomputed marks guarantee a valid mapping with respect to marks, but not necessarily an isomorphism.

First, we define the initialisation of marks for vertices by a function as

Definition 1. Let $G = (V, E)$ be an undirected graph with nodes V and edges $E = \{\{u, v\} \mid u \in V, v \in V : u \neq v\}$. Let the marks M be a possibly infinite set of distinct elements. Then, let $c : V \mapsto M$ be a function, that assigns each node $u \in V$ the same label $m \in M$ iff they are equal with respect to some specification.

Possible choices for c could be tuples comprising, for instance, the number of cycles or the length of the longest path in a nodes neighbourhood as well as the number of nodes having no connections to any other direct neighbour. Additionally, c can make use of prior information assigned to a node such as labels or statistics related to the object represented by the node. The only constraint is that any two nodes mapped to the same mark can be matched to each other by a possible isomorphism. All marks assigned to a graph form a mark distribution. The distribution of marks for a graph will later be used to determine the end of the preprocessing phase. A distribution is referred to as degenerated iff it consists of only one type of marks.

After each node was initialised by c with a mark computed, we compute higher-order marks for each node using the following recursion. The idea is that the marks are spread over edges and thus connectivity contributes to the improvement of information saved in the marks assigned to each node.

Definition 2. Let (M_i) be a family of marks, let $G = (V, E)$ be a graph and $N(u) = \{v \mid \{u, v\} \in E\}$ the neighbourhood of u (excluding u). Furthermore, let c be defined as in definition 1 and M_0 its codomain. Then the generalized recursion $g^i : V \mapsto M_i$ is defined as

$$g^0(u) \equiv c(u) \quad (3)$$

$$g^i(u) \equiv h(\{g^{i-1}(u)\} \cup \{d(g^{i-1}(v)) \mid v \in N(u)\}). \quad (4)$$

The function h is an injective function of finite multisets of marks from M_i into M_{i+1} . Furthermore, d is an injective function $d : M_i \mapsto M_i$ identifying those marks within the argument set of h_i that are travelled over edges and is referred to as distance function.

g^i can be efficiently evaluated using dynamic programming based on the memoized marks of level $i - 1$ to compute level i . The maximal recursion depth n is found when

¹ A white paper on www.itgi-algorithm.de provides a detailed runtime comparison as well as its source code.

the mark distributions for g^i and g^{i-1} are equal, i.e., their vectors of frequency counts are equal. From an information-theoretic point of view we now have maximized the mutual information between the two recursion levels. However a full-discussion of this topic is beyond the scope of this Chapter.

We next show that equal marks computed by the generalized recursion imply a possible mapping of nodes that is at least consistent with the marks assigned by g^0 and also ranges over more distant nodes. In this way, the marks of g^0 and higher recursion levels can be used as guidance information while searching for isomorphisms.

Because h and d are injective, it follows that if two nodes have equal marks, their marks and those of their neighbourhood in the prior recursion step were also equal. This represents the basis for computing a mark-consistent mapping. We formulate this observation

Lemma 1. *For any u, u^* having $g^{i+1}(u) = g^{i+1}(u^*)$ follows*

$$g^i(u) = g^i(u^*) \quad (5)$$

and

$$\forall v \in N(u) \exists v^* \in N(u^*) : g^i(v) = g^i(v^*). \quad (6)$$

Based on this, we show that equal marks allow for the computation of a mapping consistent with the g^0 marks by giving a constructive algorithm. This guarantees the computation of valid mappings during isomorphism search and therefore avoids unnecessary computations.

Theorem 1. *Let n be the actual recursion level and u, u^* be nodes with $g^n(u) = g^n(u^*)$. Then a g^0 mark-consistent mapping can be computed for all nodes v, v^* with distance n to u, u^* , respectively.*

Proof. To prove the theorem we define a constructive algorithm \mathbf{A} and its invariant I (Algorithm [1](#)). The invariant relates to an internal data structure of the algorithm consisting of triples (a, b, c) that contains the desired mapping $b \mapsto c$ and the recursion level a . We prove the theorem by showing that the invariant holds during initialization and within each iteration. Furthermore, we show that the algorithm terminates.

Initialization: In line 4 the data structure \mathcal{A} is initialized with input $(i+1, u, u^*)$. Then, the invariant holds due to $k \leftarrow i+1$ and the start nodes u and u^* have the same mark because of the input condition in line 2. Therefore, the invariant holds prior to the main loop in line 6.

Iteration: Within each loop iteration for each triple $(k, v, v^*) \in \mathcal{A}$ is tested, whether v has a neighbour $w \in N(v)$ without a triple $(a, b, c) : b = w$ in \mathcal{A} . Let w be such a neighbour, then there exists $w^* \in N(v^*) : g^{k-1}(w) = g^{k-1}(w^*)$ such that I holds for $g^k(v) = g^k(v^*)$ and Lemma [1](#). Hence, $M = M^*$. The sets M, M^* can be decomposed to

$$\begin{aligned} M &= M_{\mathcal{A}} \cup M_{\neg\mathcal{A}} \\ &= \{w \in N(v) : \exists(a, b, c) \in \mathcal{A} : b = w\} \cup \{w \in N(v) : \nexists(a, b, c) \in \mathcal{A} : b = w\} \\ M^* &= M_{\mathcal{A}^*} \cup M_{\neg\mathcal{A}^*} \\ &= \{w^* \in N(v^*) : \exists(a, b, c) \in \mathcal{A} : c = w^*\} \cup \{w^* \in N(v^*) : \nexists(a, b, c) \in \mathcal{A} : c = w^*\}. \end{aligned}$$

All nodes enter pairwise into \mathcal{A} and the invariant I holds, therefore it follows

$$(M = M^* \wedge M_{\mathcal{A}} = M_{\mathcal{A}}^*) \Rightarrow M_{\neg\mathcal{A}} = M_{\neg\mathcal{A}}^*. \quad (7)$$

And thus

$$g^{k-1}(w) = m \in M_{\neg\mathcal{A}} \Rightarrow \exists w^* \in N(v^*) : g^{k-1}(w^*) = m^* \in M_{\neg\mathcal{A}}^*. \quad (8)$$

Consequently, at any time an appropriate w^* can be chosen by \mathbf{A} and stored in \mathcal{A} as triple $(k-1, w, w^*)$ such that the invariant remains valid. Thus, I remains valid in line 10 and, therefore, in line 13. We obtain g^0 consistency using equation 5.

Termination: The algorithm \mathbf{A} terminates because $|k|$ is set to $i+1$ in line 3, is decremented in line 13 and the loop iterates while $0 \leq |k|$. I holds during each iteration, therefore it is still valid when \mathbf{A} terminates.

Having computed information about the graphs topology encoded in the marks, we can search for graph isomorphisms. To this end, we specify a backtracking Algorithm 3 that is guided by the computed marks. We improve the search by precomputing an optimized testing order for edges such that wrong assignments are found as early as possible (see Algorithm 2).

The algorithm starts by selecting a node having a mark with minimal probability and maximal degree. Then, all edges are ordered using a BFS traversal such that cross-edges are checked prior to forward edges. This assures that the search space is first tested for consistency prior expansion. Using this order, all mark-preserving relations between the graphs of both nodes are enumerated and checked for isomorphisms. Both algorithms are called from the top-level routine 4 that first checks the mark distributions for equality and degeneracy.

Algorithm 1. The algorithm $\mathbf{A}(g, i+1, u, u^*)$ for the proof of Theorem 1

```

1: {Inputs are mapping  $g$  of marks, recursion level  $i+1$ .}
2: {and two nodes  $u, u^*$  with  $g^{i+1}(u) = g^{i+1}(u^*)$ }
3:  $k \leftarrow i+1$ 
4:  $\mathcal{A} \leftarrow \{(k, u, u^*)\}$ 
5: Invariant  $I$ :  $\mathcal{A}$  contains triple  $(k, v, v^*) : g^k(v) = g^k(v^*)$  for  $0 \leq k \leq i+1$ 
6: while  $0 \leq k$  do
7:   for all  $(k, v, v^*) \in \mathcal{A}$  do
8:     for all  $w \in N(v) : \exists(a, b, c) \in \mathcal{A} : b = w$  do
9:       Let  $w^* \in N(v^*) : g^{k-1}(w) = g^{k-1}(w^*) \wedge \exists(a, b, c) \in \mathcal{A} : c = w^*$ 
10:       $\mathcal{A} \leftarrow \mathcal{A} \cup \{(k-1, w, w^*)\}$ 
11:     end for
12:   end for
13:    $k \leftarrow k-1$ 
14: end while

```

For c we generated marks that were tuples composed of the degree and label (α -helix/ β -sheet) of a node.

In our implementation of the generalized recursion we followed a semi-arithmetical idea when defining d and h . As each mark is unique and distinguishable from others, they can be used to form the basis of a number system. Therefore, we defined digits consisting of marks and coefficients as well as the addition.

Algorithm 2. `calculate-visiting-order(G, p_g)` Given a set of computed marks, this algorithm computes an optimized visiting order of the nodes of a graph such that wrong assignments are uncovered as early as possible. The idea is to explore the graph from one node by BFS and check cross-edges before forward edges. The algorithm starts with a maximal informative node having minimal probability P_c^n and maximal node degree. The symbol \bowtie represents the list concatenation operation.

```

1: Let  $p_g$  be the mark distribution for recursion  $g$  of  $G$ 
2: Let  $\text{start} \in V$  be node with  $p_g(X = \text{start})$  minimal
   and  $\text{degree}(\text{start})$  maximal
3:  $\text{queue} \leftarrow (\text{start})$ 
4:  $\text{visited}(\text{start}) \leftarrow \text{true}$ 
5:  $\text{list} \leftarrow ()$ 
6: while  $\text{not-empty}(\text{queue})$  do
7:    $\text{node} \leftarrow \text{removeFirst}(\text{queue})$ 
8:    $\text{append} \leftarrow ()$ 
9:   for  $e \in \text{AdjacentEdges}(\text{node})$  do
10:    if  $\text{not-visited}(e)$  then
11:      Let  $e = \{u, v\}$ 
12:      if  $\text{visited}(u) \wedge \text{visited}(v)$  then
13:        {append cross-edges at front}
14:         $\text{append} \leftarrow (e) \bowtie \text{append}$ 
15:      else
16:        {append forward-edges at end}
17:        if  $\text{not-visited}(u)$  then
18:           $\text{visited}(u) \leftarrow \text{true}$ 
19:           $\text{queue} \leftarrow \text{queue} \bowtie (u)$ 
20:        end if
21:        if  $\text{not-visited}(v)$  then
22:           $\text{visited}(v) \leftarrow \text{true}$ 
23:           $\text{queue} \leftarrow \text{queue} \bowtie (v)$ 
24:        end if
25:         $\text{append} \leftarrow \text{append} \bowtie (e)$ 
26:      end if
27:       $\text{visited}(e) \leftarrow \text{true}$ 
28:    end if
29:  end for
30:   $\text{list} \leftarrow \text{list} \bowtie \text{append}$ 
31: end while
32: return  $\text{list}$ 

```

Definition 3. Let M be the codomain of a c and $k \in \mathbb{N}$ any natural number. Then, a digit is denoted by $k \bullet m$ with base $m \in M$ and coefficient k . We define the addition of two digits $a \bullet m$ and $b \bullet m$ to be

$$a \bullet m + b \bullet m \equiv (a + b) \bullet m. \quad (9)$$

A number is then given by $x = \sum_{m \in M} k \bullet m$ and denoted by the term mark number. Two mark numbers x, y having coefficients $a(x), b(y)$ are added by adding corresponding coefficients.

Algorithm 3. `backtrack(list,i)` This Algorithm describes the backtracking procedure for searching isomorphisms. The data structure `iso` stores the constructed isomorphism, while `list` contains the optimized visiting order from Algorithm 2. To keep the algorithm neat $m(\cdot)$ denotes the mark assigned by g^n to a node and \perp represents an undefined value.

```

1: if size(list)=i then
2:   {recursion base}
3:   print "isomorphism found!"; STOP
4: else
5:   {recursion step}
6:   Let  $e^G = \{u^G, v^G\}$  be i.th edge from list
7:   if iso( $e^G$ ) =  $\perp$  then
8:     if non-is-visited( $u^G, v^G$ ) then
9:       {Check all edges with appropriate marks  $m(u^H), m(v^H)$ }
10:      for all  $e^H = \{u^H, v^H\} \in E : m(v^G) = m(v^H) \wedge m(u^G) = m(u^H)$  do
11:        {Each undirected edge fits this condition twice!}
12:        iso( $e^G$ )  $\leftarrow e$ ; iso( $u^G$ )  $\leftarrow u^H$ ; iso( $v^G$ )  $\leftarrow v^H$ 
13:        call backtrack(list,i+1)
14:        iso( $e^G$ )  $\leftarrow \perp$ ; iso( $u^G$ )  $\leftarrow \perp$ ; iso( $v^G$ )  $\leftarrow \perp$ 
15:      end for
16:    else if one-is-visited( $u^G, v^G$ ) then
17:      Let  $u^G$  be visited node and  $u^H$  corresponding node
18:      for all  $e^H \in \text{AdjacentEdges}(u^H) : m(v^G) = m(v^H)$  do
19:        iso( $e^G$ )  $\leftarrow e^H$ ; iso( $u^G$ )  $\leftarrow u^H$ ; iso( $v^G$ )  $\leftarrow v^H$ 
20:        call backtrack(list,i+1);
21:        iso( $e^G$ )  $\leftarrow \perp$ ; iso( $u^G$ )  $\leftarrow \perp$ ; iso( $v^G$ )  $\leftarrow \perp$ 
22:      end for
23:    else
24:      {both nodes are visited}
25:      Let  $u^H, v^H$  be the assigned nodes to  $u^G, v^G$ 
26:      if exists-edge( $u^H, v^H$ ) then
27:        Let  $e^H = \{u^H, v^H\}$  be this edge
28:        iso( $e^G$ )  $\leftarrow e^H$ ; iso( $u^G$ )  $\leftarrow u^H$ ; iso( $v^G$ )  $\leftarrow v^H$ 
29:        call backtrack(list, i + 1);
30:        iso( $e^G$ )  $\leftarrow \perp$ ; iso( $u^G$ )  $\leftarrow \perp$ ; iso( $v^G$ )  $\leftarrow \perp$ 
31:      end if
32:    end if
33:  end if
34: end if

```

$$x + y \equiv \left(\sum_{m \in M} a(x) \bullet m \right) + \left(\sum_{m \in M} b(y) \bullet m \right) = \sum_{m \in M} (a(x) + b(y)) \bullet m. \quad (10)$$

Using Definition 3, we define h to be the mark number addition, d the identity and obtain.

Algorithm 4. `graph-isomorphism(G, H)` The main procedure is designed to efficiently filter candidates that can not match. First the mark distributions for both graphs are computed and tested. The expensive backtracking step is only carried out if the distributions are not degenerate and are equal. In the degenerated case a real implementation should warn or switch to another implementation of c . Only if no degenerate distributions for both graphs are available the expensive backtracking search is carried out using the optimized visiting order.

```

1: Calculate converged  $g_G^n, g_H^n$  mark distributions
2: if Mark distributions are not degenerated and are equal then
3:   list  $\leftarrow$  calculate-visiting-order( $G, p_G^n$ )
4:   call backtrack(list,0)
5: end if
6: print “G and H are not isomorph”

```

Definition 4. Let $G = (V, E)$ be a graph, $u \in V$ any node and $c : V \mapsto M$. Then the mark number recursion is defined by

$$g^0(u) \equiv 1 \bullet c(u) \tag{11}$$

$$g^i(u) \equiv g^{i-1}(u) + \sum_{v \in N(u)} g^{i-1}(v). \tag{12}$$

This definition efficiently generates new marks as needed.

The presented generalized variant of the Weisfeiler-Lehmann algorithm facilitates subsequent graphlet analysis for a collection of graphs in three ways: First, it allows for the precomputation of highly informative marks for each graph. Those precomputed marks prevent unnecessary graph comparisons and hence reduce runtime in the case of unequal graphlets. Secondly, by construction on the other hand they facilitate the computation of an isomorphism by narrowing the search space. And third, they allow for the detection of degenerate graphs, i.e., those graphs that produce a degenerate mark distribution providing no structural connectivity information. In this case the implementation should switch to another definition for c (e.g. rely on more computational intensive marks based on Breath-First-Search or Centrality metrics).

Although our presented method brings in practical advantages (e.g. easy to implement, practical performance), it does not implicate any complexity statements about the GI problem itself. Users should be aware of that we have presented a practical heuristic for which degenerate cases exist, but seldom occur in real world problems. From an information-theoretic perspective it can be shown that the presented recursion for computing marks maximizes the mutual information between two subsequent mark distributions. Hence, we can assure that generated marks can not be improved any more, except by changing to another c . A full treatment of this topic however is beyond the scope of this Chapter. We thus proceed to the description of the high-level graphlet analysis in the next subsection.

Graphlet Analysis

Graphlet analysis makes use of subgraph sampling and, therefore, relies on a graph isomorphism test, presented in the previous section. Each sampled subgraph is referred to

as graphlet. Its frequency or probability within a network is estimated by repeated sampling. In addition, statistical graphlet analysis requires the knowledge of a background distribution to compute the probability of an observation. As no analytical distribution function for graphlets is known, their probabilities are in general estimated from random graphs.

To obtain a random model resembling the input graph distribution, each protein graph is randomized. We used a random rewiring method where each edge is split into two half-edges. Then, all half-edges are randomized and rewired. This is repeated until a connected graph is obtained or a maximum number of iterations is reached. In the latter case, the last sample is saved. In summary, random rewiring conserves important graph properties (e.g. the node degree). By randomizing each graph once, we obtain a collection of random graphs that closely resembles the test distribution.

Next, we estimate the graphlet distribution by random sampling connected subgraphs. The goal of the sampling is twofold: First, all existent graphlets should be detected and, second, their distribution should be estimated correctly. If all graphlets were known in advance, drawing a fixed number of samples would yield the maximum likelihood estimate of the graphlet distribution, which is a multinomial distribution [13,14]. To achieve this estimate, we employ a two-pass approach.

In the first pass, the data is exploratory sampled. For counting the graphlet frequencies, we make use of a Move-to-Front (MF) list that holds a counter for each graphlet type. We need the graph isomorphism algorithm when searching this list. While sampling, each sampled graphlet is first searched in the MF list for counting its occurrence and inserted in the case it is not found. Therefore, the MF list length increases during this pass. We draw 1000 samples per graph of the database to minimize the possibility of missing patterns.

In the second pass, we keep the MF list fixed during sampling to compute the maximum likelihood estimates. Again, we draw a total of 1000 samples, 5 repetitions with 200 samples, from each graph and, thus, obtain 5 independent distribution estimates. If sampling detects an unknown graphlet within the second pass, a counter for unknown patterns is increased. Finally, we compute the distribution estimate by averaging and normalizing all samplings for a graph.

We choose a sampling size of 1000 graphlets in each phase because only a small fraction of the RNG comprises more than 40 nodes, i.e., only a small fraction of the database proteins are large. As the number of graphlets increases exponentially with pattern size, a trade-off between estimation accuracy and runtime is required. We found that a sampling size of 1000 is sufficient to accurately estimate the graphlet distribution within a reasonable runtime.

To determine significant graphlets, we employ permutation testing, which is a non-parametric technique to efficiently test the equality of two distributions g, h [15]. It requires no assumptions on the true probability distributions. By resampling new random distributions from the input distribution and evaluating a test statistic T , it infers the distribution of T under $H_0 : g = h$. This denotes the case where g and h are equal and, therefore, allows to compute the p-value for the hypothesis that the input distributions are equal. In our experiments, we use the `coin` package implemented in R [16].

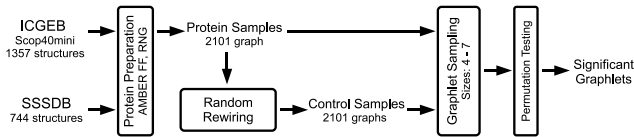


Fig. 1. The graphlet analysis workflow using a random control and permutation testing

Table 1. Results of the Graphlet Analysis

Graphlet Size	Found Graphlets	Significant Graphlets	Filter Ratio
4	427	22	0.05
5	1,731	77	0.04
6	5,366	212	0.04
7	16,904	316	0.02

For each graphlet, we employ permutation testing to compute the likelihood that its distribution on the protein graphs equals its distribution on the randomized graphs. Hence, we determine those graphlets that are significantly over or under-represented in random graphs. To account for the multiple-testing problem, we afterwards adjust the p-values according to the method of [17]. Finally, all graphlets below a significance threshold of 0.05 and having a frequency ratio at least 10-fold larger on protein graphs than on the random graphs are retained for further analysis. Figure 1 illustrates this analysis step.

Decision Tree Analysis

Here, we relate the extracted graphlets to global protein structure properties, e.g. the SCOP classification of each structure.

Therefore, we employ decision trees as they facilitate a better understanding of the data set. Inspection of leafs and their associated samples allows relating specific graphlets to proteins and, thus, facilitates the analysis of structures with respect to interaction patterns. In addition, decision trees are native multi-class prediction algorithms and are thus a convenient choice for the prediction of various protein classes. However, decision trees tend to overfit and therefore require careful complexity regularization.

An alternative to decision trees are neural network predictors. Neural networks represent a non-linear class of regression functions that are estimated by gradient descent. For each class a single output neuron can be integrated into the network and, thus, a single neural network can model the posterior probability of each class using a common basis of hidden neurons. However, neural networks are pure highly non-linear prediction models and, therefore, hard to interpret. Consequently, they are less suited for data mining purposes and do not provide much insight into the dataset.

Using the graphlet distributions, we encode each protein as a binary vector of graphlet occurrences. Whenever a graphlet is found for a protein, it is encoded as 1 and as 0 else. In addition, we integrate the SCOP level information from ICGEB as well as the SSSDB Motif class information as a prediction target. SCOP has four hierarchical

levels: the class, fold, superfamily and family of a protein domain. Similarly, the SSSDB Motif Class encodes the super-secondary structure super class, while the SSSDB Motif Subclass denotes a more precise subdivision.

We use the `partition` platform from JMP [18] for learning. JMP is a reduced version of the SAS Enterprise Miner software suite, which is extensively used for professional data analysis in industry, and provides various platforms that facilitate data analysis tasks. Especially, the ability of the `partition` platform to easily inspect each prediction node is helpful during the analysis of the inferred graphlets.

In our experiments, the Minimum Split Size (MSS) parameter is chosen to be 5, 10, 15, 20, 25 and 50, while training on graphlets of size 4 to 7. For each model, the prediction performance is evaluated using 5-fold cross-validation to compute AUC values for each class. Then, all AUC values are averaged to yield the mean AUC (mAUC) for each combination of graphlet size and MSS. For comparison purposes, we also employ the `neural_nets` platform.

Regularization

However, to obtain regularized decision trees, we follow the work of [19]. There, the space of decision tree hypotheses is regularized by their number of leafs and their prediction error. Let \mathcal{T} be the hypothesis space of decision trees and $|T|$ denote the number of leafs of a tree T . Furthermore, let n be the number of data samples and λ be a weighting factor between tree complexity and prediction error, then the *Complexity Regularization CR* for decision tree selection is given as

$$CR = \arg \min_T \left\{ \underbrace{\hat{R}}_{\text{Empirical risk}} + \lambda \underbrace{\sqrt{\frac{|T|}{n}}}_{\text{Bound on bias}} \right\} \quad (13)$$

where \hat{R} denotes the prediction error and n the number of training samples. In our case, we set $\hat{R} = 1 - mAUC$, because we are dealing with multi-class predictions and the mAUC is a performance measure between 0 and 1 that can be converted into an error by subtracting it from 1. This error is then compared to its theoretical bound, the *Uniform Deviation Bound UDB_n* , which upper bounds the true risk of a decision tree and is given as

$$UDB_n = \arg \min_{T \in \mathcal{T}} \left\{ \hat{R}(T) + \sqrt{\frac{3|T| + \log(n)}{2n}} \right\}. \quad (14)$$

The remaining task is to find a valid λ for regularization. We choose λ to minimize

$$\lambda = \arg \min_{\lambda} \sum_T (CR - UDB_n)^2 \quad (15)$$

for all considered decision trees, trained with MSS of 5, 10, 15, 20, 25, and 50.

3 Results and Discussion

Graphlet Extraction

We convert 1357 proteins from the ICGB SCOP40mini database and 744 proteins from the SSSDB into the described graphical representation. Figure 2 illustrates the result of a conversion. This results in a total of 2101 graphs left for graphlet analysis. Two other datasets from ICGB, PCB00020 (11,944 structures from SCOP95) and PCB00026 (11,373 structures from CATH95), are convenient for our experiments. However, the enormous number of graphlets found in this data sets leads to infeasible computation times. Therefore, we restrict our analysis to the smallest ICGB collection and included the SSSDB instead.

Table 1 summarizes our sampling results. It shows that increasing the graphlet pattern size also increases the number of detected graphlets as well as the number of significant graphlets. However, the ratio between significant and detected graphlets remains below 5%. Hence, we extract only a small amount of significant patterns from a large collection of graphlets (see Table 1).

Decision Tree Learning

For each target variable the decision tree minimizing $CR - UDB_n$ is selected as the regularized model and saved for final analysis. We find that $\lambda = 1.3$ fulfils equation (15) for each target variable. We choose λ such that the total squared deviation from the UDB_n was minimal when averaged over all models using any size of graphlets and any number of splits. Consequently, all models are regularized and have maximal expressive power along with a reduced probability of overfitting. Table 2 lists the MSS, graphlet size, number of splits, as well as the mAUC of the final models for each target variable. We find that simple models suffice to achieve mAUC better than 0.82. In addition, at least 20 samples can be summarized in a leaf to result in a decision tree predicting one category without sacrificing prediction performance. Finally, we find that graphlets of size 5 are in most cases superior to others. Also graphlets of size 4 and 7 are used for decision tree learning and, therefore, provide useful information (Table 2).

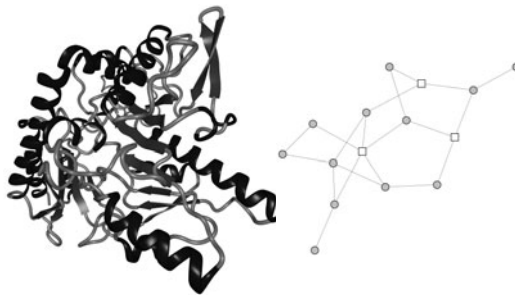


Fig. 2. Structure (left) and RNG (right) of ICGB structure d1pmma. Box nodes denote β -sheets, whereas round nodes refer to α -helices.

Table 2. Regularized Decision Trees for each target variable

Target Variable	# classes	Graphlet Size	MSS Size	Performed Splits	mAUC	Neural Net mAUC
SCOP class	5	5	20	23	0.87	0.92
SCOP fold	19	5	20	24	0.87	0.84
SCOP superfamily	7	4	20	24	0.85	0.83
SCOP family	48	5	20	16	0.83	0.87
SSSDB Motif Class	32	7	10	22	0.93	0.99
SSSDB Motif Subclass	153	5	15	21	0.96	0.96

We also compare the performance of decision trees to the performance of neural networks, also implemented in JMP. The last column of Table 2 shows the mean AUC achieved by the `neural_nets` platform using cross-validation on a random selection of 40% of the samples, which are held out as external validation set. The platform was trained using the default values. We find that the neural network is superior to our regularized decision trees in predicting the SCOP class and SCOP family, as well as the SSSDB Motif Class. However, the performance difference does not exceed a value of 0.06 mAUC points. Consequently, decision trees and neural networks achieve a comparable performance and, therefore, support the descriptive value of our graphlet features.

Next, we extract all graphlets from the selected regularized decision tree models to analyse their usages with respect to all target variables. Here, we find graphlets that are used several times for the prediction of various target variables, while other graphlets are specific to one class.

While 66 graphlets are used only for the prediction of one target variable, we find two graphlets of size 5 that are used to predict 4 different target variables (Figure 3). Note that this is the maximum number achievable because two target variables are predicted using graphlets of size 4 and 7.

In Figure 4 all SCOP superfamily graphlets are shown. Within the decision tree for the SCOP superfamily, the graphlets (a) and (b) in Figure 4 are more relevant for classification, because they are used in three and two splits, respectively. Interestingly, (a) and (b) are paths of helices containing one or two sheets, while the other graphlets consist mainly of star topologies.

Finally, we predict all detected graphlets from secondary structure sequence. Therefore, we convert each protein sequence into strings of symbols h,s,l encoding whether a residue is within a helix, strand or loop. We design a set of 439 features describing lengths, as well as densities, normalized to protein length, of helices, strands, loops. In addition, we design regular expressions for SSE sequence patterns, which result in binary features. Then, we train decision trees, implemented in the `rpart` package of R, and use 5-fold cross-validation using the `bootstrap` package to estimate the MCC [20]. We find that chains of SSEs and star topologies can be best predicted by decision trees.

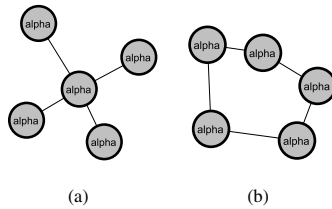


Fig. 3. Graphlet 3(a) predicts 4 targets and is predictable with $MCC=0.6$, while graphlet 3(b) predicts 4 targets and is predictable with $MCC=0.5$ from secondary-structure information

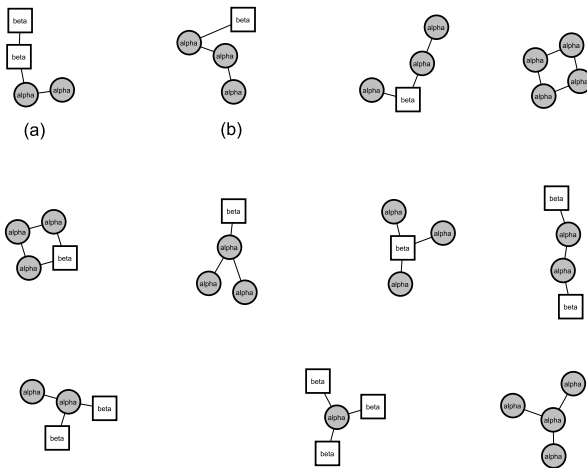


Fig. 4. This figure shows all extracted graphlets used for the prediction of the SCOP superfamily

4 Conclusions

In summary, we find that significantly overrepresented patterns in energetic SSE interactions exist and can be found using graphlet analysis. Regularized decision tree learning on the mined patterns predicts SCOP levels and SSSDB Motifs with great accuracy ($mAUC > 0.8$) using about 20 graphlets. Also, the presence of a specific graphlet can be predicted from secondary structure sequence of a protein with MCC values up to 0.7. Finally, we demonstrate that the combination of graphlet analysis using permutation testing and decision tree learning facilitates automatic categorization of protein structures.

We have shown that graphlets are predictable from the secondary structure sequence, therefore graphlets can be used as constraints for the placement of predicted secondary structure elements, when predicting the tertiary structure from the protein sequence alone. Thus, future work should focus on the usage of the predictable graphlets to improve *ab initio* protein structure prediction.

References

1. Murzin, A.G., Brenner, S.E., Hubbard, T., Chothia, C.: Scop: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology* 247, 536–540 (1995)
2. Orengo, C.A., Michie, A.D., Jones, S., Jones, D.T., Swindells, M.B., Thornton, J.M.: CATH—a hierarchic classification of protein domain structures. *Structure* 5, 1093–1108 (1997)
3. Vacic, V., Iakuoucheva, L., Lonardi, S., Radivojac, P.: Graphlet kernels for prediction of functional residues in protein structures. *Journal of Computational Biology* 17, 55–72 (2010)
4. Graphlet data mining of energetical interaction patterns in protein 3D structures. In: *International Conference on Neural Computation (ICNC)* (2010)
5. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network Motifs: Simple Building Blocks of Complex Networks. *Science* 298, 824–827 (2002)
6. Sonogo, P., Pacurar, M., Dhir, S., Kertesz-Farkas, A., Kocsor, A., Gaspari, Z., Leunissen, J.A.M., Pongor, S.: A Protein Classification Benchmark collection for machine learning. *Nucleic Acids Res.* 35, D232–D236 (2007)
7. Chiang, Y.S., Gelfand, T.I., Kister, A.E., Gelfand, I.M.: New classification of supersecondary structures of sandwich-like proteins uncovers strict patterns of strand assemblage. *Proteins* 68, 915–921 (2007)
8. Kohlbacher, O., Lenhof, H.P.: BALLrapid software prototyping in computational molecular biology. *Bioinformatics* 16, 815–824 (2000)
9. Kabsch, W., Sander, C.: Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* 22, 2577–2637 (1983)
10. Toussaint, T.G.: The relative neighbourhood graph of a finite planar set. *Pattern Recognition* 12, 261–268 (1980)
11. Milligan, G.W., Isaac, P.D.: The validation of four ultrametric clustering algorithms. *Pattern Recognition* 12, 41–50 (1980)
12. Weisfeiler, R. (ed.): *On Construction and Identification of Graphs*. Number 556. *Lecture Notes in Math*. Springer (1976)
13. Wassermann, L.: *All of statistic*. Springer (2004), theorem 14.5.
14. Georgii, H.O.: *Stochastik*, 2nd edn., p. 198. de Gruyter (2004)
15. Wald, A., Wolfowitz, J.: Statistical tests based on permutations of the observations. *The Annals of Mathematical Statistics* 15, 358–372 (1944)
16. R Development Core Team: *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria (2009) ISBN 3-900051-07-0
17. Holm, S.: A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* 6, 65–70 (1979)
18. SAS Institute Inc.: *Jmp 8.0.1* (2009), <http://www.jmp.com>
19. Scott, C., Nowak, R.: On the adaptive properties of decision trees. In: *Advances in Neural Information Processing Systems*, vol. 17. MIT Press (2005)
20. Baldi, P., Brunak, S., Chauvin, Y., Andersen, C.A.F., Nielsen, H.: Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics* 16, 412–424 (2000)

Learning from Data as an Optimization and Inverse Problem

Věra Kůrková

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, Prague 8, Czech Republic
vera@cs.cas.cz

Abstract. Learning from data is investigated as minimization of empirical error functional in spaces of continuous functions and spaces defined by kernels. Using methods from theory of inverse problems, an alternative proof of Representer Theorem is given. Regularized and non regularized minimization of empirical error is compared.

Keywords: Learning from data, Minimization of empirical error, Inverse problems, Reproducing kernel Hilbert spaces.

1 Introduction

Supervised learning can be formally described as an optimization problem of minimization of error functionals over parameterized sets of input-output functions computable by a given computational model. Various learning algorithms iteratively modify parameters of the model until sufficiently small values of error functionals are achieved and the corresponding input-output function of the model fits sufficiently well to the training data (see, e.g., [12]).

Error functionals are determined by training data and loss functions. Training data are sets of pairs of inputs and outputs. A loss function measures how much is lost when the system computes $f(x)$ instead of y . The most common loss function is the quadratic loss which has been used over two hundred years since the invention of the least square method by Gauss and Legendre.

Traditionally, the least square method has been applied to finding among elements of linear sets of functions those best fitting to data. In distributed connectionistic computational models (such as neural and kernel networks), best fitting functions are searched for in non linear sets of input-output functions. Learning algorithms optimize besides of coefficients of linear combinations of computational units also their inner parameters (such as input weights, centers, biases, widths).

Such algorithms in their best only achieve good fit to the training data. It was proven that for typical computational units such as sigmoidal perceptrons and Gaussian kernels, sufficiently large network can interpolate a given training data exactly, i.e., error functionals can achieve the value zero [34]. However, networks perfectly fitting to randomly chosen noisy training samples may be too much influenced by the noise and may not perform well on data that were not chosen for training. Thus various attempts to modify error functionals to improve so called “generalization capability” has been proposed.

Girosi and Poggio [5] introduced into learning theory a method of regularization as a means of improving generalization. They considered modifications of error functionals based on Tikhonov regularization which adds to an error functional an additional functional, called stabilizer, penalizing undesired properties of input-output functions such as high-frequency oscillations [6]. In practical applications, various simple stabilizers have been successfully used, e.g., seminorms based on derivatives [7] or sums of squares or absolute values of output weights (so called output weight regularization) [12].

Girosi, Jones and Poggio [6] characterized minima of empirical errors regularized by stabilizers in the form of squares of weighted Fourier transforms over computational model with kernel units. Later, Girosi [8] found that these stabilizers belong to a wider class of functionals formed by squares of norms on a special class of function spaces called reproducing kernel Hilbert spaces (RKHSs). These spaces were defined by Aronszajn [9], introduced to data analysis by Parzen [10], and applied to data interpolation by splines by Wahba [11]. RKHSs became popular in soft-computing due to the use of kernels in classification algorithm called support vector machine (SVM) [12][13] (see, also, [14]). A unifying framework for theory of learning in RKHSs as ambient function spaces was presented in [15].

Regularization was developed in 1970s as a method of improving stability of solutions of certain problems from physics called inverse problems, where unknown causes (shapes of functions, forces or distributions) of known consequences (measured data) have to be found. These problems has been studied in applied science, such as acoustics, geophysics and computerized tomography (see, e.g., [16]). To solve such a problem, one needs to know how unknown causes determine known consequences, which can often be described in terms of an operator. In problems originating from physics, dependence of consequences on causes is usually described by integral operators (such as those defining Radon or Laplace transforms [17][18]). Kůrková [19][20] and DeVito et al. [21] represented minimization of error functionals with quadratic loss function as inverse problems defined by evaluation and inclusion operators. Theory of inverse problems provides tools for characterization of solutions of problems defined by continuous operators with Hilbert spaces as their domains and ranges. As RKHSs satisfy this condition, theory of inverse problems can be applied to minimization of error functionals over RKHSs.

In this paper, we investigate learning as minimization of error functionals over RKHSs and over spaces of continuous functions. We survey and extend results on applications of theory of inverse problems to minimization of empirical error functional with the quadratic loss. Using methods from theory of inverse problems, we give an alternative proof of so called “Representer Theorem” [15] holding also for non continuous kernels, and show that regularized solutions of the minimization task differ from a non regularized ones merely in coefficients of linear combinations of computational units. Combining characterization of optimal solutions of learning tasks over RKHSs with density results holding for Gaussian kernel networks with fixed widths, we describe a class of argminima of empirical error functional over the whole space of continuous functions.

The paper is organized as follows. Section 2 presents some properties of empirical error functional and its representation as a distance functional. In section 3, basic

terminology and results from theory of inverse problems are recalled and minimization of empirical error functional is reformulated as an inverse problem. In section 4 these tools are applied to description of theoretically optimal input-output functions in learning from data over spaces defined by kernels. In section 5, their consequences to minimization of empirical error over spaces of continuous functions are derived.

2 Properties of Empirical Error

Supervised learning can be theoretically studied as an optimization problem of minimization of a suitable error functional defined by a training sample over a set of input-output functions of a given computational model. The most widespread functional minimized by various learning algorithms (such as back-propagation or genetic algorithms) is called an *empirical error*. It is determined by a sample $z = \{(u_i, v_i) \in X \times Y \mid i = 1, \dots, m\}$, where $X \subseteq \mathbb{R}^d$ and $Y \subseteq \mathbb{R}$, of input-output pairs of data and a loss function V . The empirical error functional is denoted $\mathcal{E}_{z,V}$ and defined as

$$\mathcal{E}_{z,V}(f) = \frac{1}{m} \sum_{i=1}^m V(f(u_i), v_i).$$

We denote by \mathcal{E}_z the *empirical error with the quadratic loss function*, i.e.,

$$\mathcal{E}_z(f) = \frac{1}{m} \sum_{i=1}^m (f(u_i) - v_i)^2. \tag{1}$$

One of many advantages of the quadratic loss function is that it enables to reformulate a minimization of empirical error as minimization of a weighted Euclidean distance. For a sample $z = ((u_1, v_1), \dots, (u_m, v_m))$, let $u = (u_1, \dots, u_m)$, $v = (v_1, \dots, v_m)$, and $\|\cdot\|_{2,m}$ denote the weighted ℓ^2 -norm on \mathbb{R}^m defined as

$$\|x\|_{2,m} = \sqrt{\frac{1}{m} \sum_{i=1}^m x_i^2}.$$

Then for every $f : X \rightarrow \mathbb{R}$ we have

$$\mathcal{E}_z(f) = \|(f(u_1), \dots, f(u_m)) - (v_1, \dots, v_m)\|_{2,m}^2. \tag{2}$$

So minimization of the empirical error \mathcal{E}_z over any space of functions on $X \subseteq \mathbb{R}^d$ such that $\{u_1, \dots, u_m\} \subseteq X$ is equivalent to minimization of the $\|\cdot\|_{2,m}$ -distance between the vector of the output data $v = (v_1, \dots, v_m)$ and the vector $(f(u_1), \dots, f(u_m))$ obtained by evaluating the function f at the input data $u = (u_1, \dots, u_m)$.

Learning algorithms aim to minimize error functional over hypothesis sets of input-output functions of suitable computational models. These model consist of units, which can be formally described as functions of two variables

$$\phi : X \times A \rightarrow \mathbb{R},$$

where $X \subseteq \mathbb{R}^d$ is a set of inputs and $A \subseteq \mathbb{R}^s$ is a set of parameters. The set

$$G_\phi(A) = \{\phi(\cdot, a) \mid a \in A\}$$

is called a dictionary. The most widespread computational model used in soft-computing is a one-hidden-layer network with a linear output. Such network computes input-output functions from the set

$$\text{span}_n G_\phi(A) = \left\{ \sum_{i=1}^n w_i \phi(\cdot, a_i) \mid w_i \in \mathbb{R}^d, a_i \in A \right\}.$$

Many classes of neural networks have so called “universal approximation property” which is defined as density of sets of input-output functions in spaces $(\mathcal{C}(X), \|\cdot\|_{\text{sup}})$ of continuous functions with the supremum norm $\|f\|_{\text{sup}} = \sup_{x \in X} |f(x)|$ or in $(\mathcal{L}^p(X), \|\cdot\|_p)$ with X compact. So for a rich variety of hidden unit functions ϕ , the sets $\text{span } G_\phi(A) = \bigcup_{n=1}^\infty \text{span}_n G_\phi(A)$ are dense in $(\mathcal{C}(X), \|\cdot\|_{\text{sup}})$ for all $X \subset \mathbb{R}^d$ compact.

It is easy to show that for any sample z , the empirical error \mathcal{E}_z is continuous on $(\mathcal{C}(X), \|\cdot\|_{\text{sup}})$.

Proposition 1. *Let $X \subseteq \mathbb{R}^d$, $z = (u, v)$, where $u = (u_1, \dots, u_m) \in X^m$ and $v = (v_1, \dots, v_m) \in \mathbb{R}^m$. Then \mathcal{E}_z is continuous on $(\mathcal{C}(X), \|\cdot\|_{\text{sup}})$.*

Proof. Let $f, h \in \mathcal{C}(X)$ be such that $\|f - h\|_{\text{sup}} < \delta$. Then

$$|\mathcal{E}_z(f) - \mathcal{E}_z(h)| = \frac{1}{m} \left| \sum_{i=1}^m ((f(u_i) - h(u_i))(f(u_i) + h(u_i) - 2v_i)) \right| \leq \delta(C + m\delta),$$

where $C = \max_{i=1, \dots, m} 2f(u_i)$. So \mathcal{E}_z is continuous at f . □

So for ϕ continuous, \mathcal{E}_z is also continuous on the linear subspace $\text{span } G_\phi(A)$ of $\mathcal{C}(X)$. The next proposition shows that any argminimum of a continuous functional over a dense subset is also an argminimum over the whole space.

Proposition 2. *Let \mathcal{Y} be a dense subset of a normed linear space $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$, $T : (\mathcal{X}, \|\cdot\|_{\mathcal{X}}) \rightarrow \mathbb{R}$ be a continuous functional, and $f \in \mathcal{Y}$ be an argminimum of T over \mathcal{Y} . Then f is an argminimum of T over \mathcal{X} .*

Proof. Assume by contradiction that there exists $g \in \mathcal{X}$ such that $T(g) < T(f)$. Let $\eta > 0$ be such that $T(g) + \eta < T(f)$. By continuity of T at g , there exists $\delta > 0$ such that for all $g' \in \mathcal{X}$, $\|g - g'\|_{\mathcal{X}} < \delta$ implies $|T(g) - T(g')| < \eta$. By density of \mathcal{Y} , there exist $g' \in \mathcal{Y}$ with $\|g - g'\|_{\mathcal{X}} < \delta$ and so $|T(g) - T(g')| < \eta$. As $g' \in \mathcal{Y}$, we have $T(g) + \eta < T(f) \leq T(g')$ which gives a contradiction. □

Thus for any dictionary $G_\phi(A)$ for which $\text{span } G_\phi(A)$ is dense in $(\mathcal{C}(X), \|\cdot\|_{\text{sup}})$, an argminimum of \mathcal{E}_z over $\text{span } G_\phi(A)$ is also an argminimum over $\mathcal{C}(X)$.

3 Minimization of Empirical Error as an Inverse Problem

The representation (2) of empirical error with the quadratic loss function as a distance functional allows us to apply to learning theory results from theory of inverse problems.

For a linear operator $A : (\mathcal{X}, \|\cdot\|_{\mathcal{X}}) \rightarrow (\mathcal{Y}, \|\cdot\|_{\mathcal{Y}})$ between two Hilbert spaces (in finite-dimensional case, a matrix A) an *inverse problem* determined by A is to find for $g \in \mathcal{Y}$ (called *data*) some $f \in \mathcal{X}$ (called *solution*) such that

$$A(f) = g$$

(see, e.g., [17]).

To reformulate minimization of \mathcal{E}_z as an inverse problem, define for the input sample $u = (u_1, \dots, u_m) \in \mathbb{R}^m$ and any space \mathcal{S} of functions on some $X \subseteq \mathbb{R}^d$ such that $\{u_1, \dots, u_m\} \subseteq X$ an *evaluation operator* $J_u : \mathcal{S} \rightarrow \mathbb{R}^m$ as

$$J_u(f) = (f(u_1), \dots, f(u_m)). \quad (3)$$

The representation (2) implies that

$$\mathcal{E}_z(f) = \|J_u(f) - v\|_{2,m}^2. \quad (4)$$

Thus minimizing the empirical error \mathcal{E}_z with the quadratic loss function over \mathcal{S} is equivalent to solving an *inverse problem given by the evaluation operator J_u for the data v* .

An inverse problem is called *well-posed* if for every $g \in \mathcal{Y}$ there exists a unique solution $f \in \mathcal{X}$, which depends continuously on data. So by the Banach open map theorem [22], for a well-posed inverse problem there exists a unique continuous inverse operator $A^{-1} : \mathcal{Y} \rightarrow \mathcal{X}$. However, a continuous dependence of solutions on data may not always guarantee robustness against a noise. A *condition number* defined for a well-posed problem given by an operator A as $\text{cond}(A) = \|A\| \|A^{-1}\|$ measures stability. Problems with large condition numbers are called *ill-conditioned*.

When for some $g \in \mathcal{Y}$ no solution exists, at least one can search for a *pseudosolution* f° , for which $A(f^\circ)$ is a best approximation to g among elements of the range of A , i.e.,

$$\|A(f^\circ) - g\|_{\mathcal{Y}} = \min_{f \in \mathcal{X}} \|A(f) - g\|_{\mathcal{Y}}.$$

The unique pseudosolution with minimal norm $\|\cdot\|_{\mathcal{X}}$ is called *normal pseudosolution* and denoted f^+ .

Theory of inverse problems overcomes ill-conditioning by using various regularized solutions. Tikhonov's regularization [23] replaces the problem of minimization of the functional $\|A(\cdot) - g\|_{\mathcal{Y}}^2$ with minimization of

$$\|A(\cdot) - g\|_{\mathcal{Y}}^2 + \gamma \Psi,$$

where Ψ is a functional called *stabilizer* and the *regularization parameter* γ plays the role of a trade-off between an emphasis on a proximity to data and a penalization of undesired solutions expressed by Ψ . A typical choice of a stabilizer is the square of the norm on \mathcal{X} , $\|\cdot\|_{\mathcal{X}}^2$, for which Tikhonov regularization minimizes the functional

$$\|A(\cdot) - g\|_{\mathcal{Y}}^2 + \gamma \|\cdot\|_{\mathcal{X}}^2. \quad (5)$$

Originally, properties of pseudoinverse and regularized inverse operators were described for operators between finite dimensional spaces, where such operators can be represented by matrices [24,25]. In 1970s, the theory of pseudoinversion was extended to infinite-dimensional spaces – it was shown that similar properties as the ones of Moore-Penrose pseudoinverses of matrices also hold for pseudoinverses of *continuous linear operators* between Hilbert spaces [26]. Continuous operators have *adjoint operators* $A^* : \mathcal{Y} \rightarrow \mathcal{X}$ satisfying the equation

$$\langle A(f), g \rangle_{\mathcal{Y}} = \langle f, A^*(g) \rangle_{\mathcal{X}}$$

[22]. The adjoints play an important role in a characterization of pseudosolutions and regularized solutions.

The next paragraph summarizes basic results from theory of inverse problems from [17, pp.68-70] and [26, pp.74-76]. For every *continuous linear operator* $A : (\mathcal{X}, \|\cdot\|_{\mathcal{X}}) \rightarrow (\mathcal{Y}, \|\cdot\|_{\mathcal{Y}})$ between two Hilbert spaces there exists a unique continuous linear pseudoinverse operator $A^+ : \mathcal{Y} \rightarrow \mathcal{X}$ when the range $R(A)$ is closed, otherwise A^+ is defined only for those $g \in \mathcal{Y}$, for which the projection $\pi_{clR(A)}(g) \in R(A)$. The pseudoinverse A^+ satisfies for every $g \in \mathcal{Y}$,

$$\|A^+(g)\|_{\mathcal{X}} = \min_{f^o \in S(g)} \|f^o\|_{\mathcal{X}},$$

where $S(g) = \operatorname{argmin}(\mathcal{X}, \|A(\cdot) - g\|_{\mathcal{Y}})$, and for every $g \in \mathcal{Y}$, $AA^+(g) = \pi_{clR(A)}(g)$ and

$$A^+ = (A^*A)^+ A^* = A^*(AA^*)^+. \tag{6}$$

For every $\gamma > 0$, there exists a unique operator

$$A^\gamma : \mathcal{Y} \rightarrow \mathcal{X}$$

such that for every $g \in \mathcal{Y}$,

$$\{A^\gamma(g)\} = \operatorname{argmin}(\mathcal{X}, \|A(\cdot) - g\|_{\mathcal{Y}}^2 + \gamma\|\cdot\|_{\mathcal{X}}^2)$$

and

$$A^\gamma = (A^*A + \gamma I_{\mathcal{X}})^{-1} A^* = A^*(AA^* + \gamma I_{\mathcal{Y}})^{-1} \tag{7}$$

where $I_{\mathcal{X}}, I_{\mathcal{Y}}$ denote the identity operators. For every $g \in \mathcal{Y}$, for which $A^+(g)$ exists,

$$\lim_{\gamma \rightarrow 0} A^\gamma(g) = A^+(g). \tag{8}$$

4 Minimization of Empirical Error over RKHSs

The representation (4) of minimization of an empirical error functional as an inverse problem defined by an evaluation functional offers a possibility of characterization of argminima of empirical error. However, properties of solutions of inverse problems (6), (7), and (8) were derived under an assumption that the operator defining the inverse problem is a *continuous operator between Hilbert spaces*. Aronszajn [9] introduced a

class of function spaces called *Reproducing Kernel Hilbert Spaces* (RKHSs) as Hilbert spaces of point-wise defined functions on which *all evaluation functionals are continuous*. His definition implies that a RKHS is uniquely determined by a symmetric positive semidefinite kernel $K : X \times X \rightarrow \mathbb{R}$, i.e., K satisfying for all n and all n -tuples $\{x_1, \dots, x_m\} \subseteq X$ and all $\{a_1, \dots, a_m\} \subset \mathbb{R}$,

$$\sum_{i,j=1}^n a_i a_j K(x_i, x_j) \geq 0.$$

A RKHS determined by K , denoted $\mathcal{H}_K(X)$, contains all linear combinations of functions of the form $K_x : X \rightarrow \mathbb{R}, x \in X$, defined as $K_x(y) = K(x, y)$ (these functions are called *representers* and limits of Cauchy sequences in the norm $\|\cdot\|_K$ induced by the inner product defined on representers as $\langle K_x, K_y \rangle_K = K(x, y)$). An important feature of RKHSs is so called *reproducing property* which guarantees that for all $f \in \mathcal{H}_K(X)$ and all $x \in X$

$$\langle f, K_x \rangle_K = f(x). \tag{9}$$

A paradigmatic example of a kernel is the *Gaussian kernel* $K(x, y) = e^{-\|x-y\|^2}$. A RKHS defined by the Gaussian kernel contains all linear combinations of translations of Gaussians, so it contains input-output functions of radial-basis-function networks with the Gaussian radial function with a fixed width equal to one.

Continuity of evaluation functionals on RKHSs allows application of results from theory of inverse problems to minimization of empirical error over kernel networks. First, we describe properties of evaluation operators on RKHSs.

Proposition 3. *Let $X \subseteq \mathbb{R}^d, K : X \times X \rightarrow \mathbb{R}$ be a symmetric positive semidefinite kernel. Then for every positive integer m and every $u \in X^m$*

- (i) $J_u : (\mathcal{H}_K(X), \|\cdot\|_K) \rightarrow (\mathbb{R}^m, \|\cdot\|_{2,m})$ is continuous;
- (ii) $R(J_u)$ is closed in $(\mathbb{R}^m, \|\cdot\|_{2,m})$;
- (iii) J_u is compact;
- (iv) the adjoint $J_u^* : (\mathbb{R}^m, \|\cdot\|_{2,m}) \rightarrow (\mathcal{H}_K(X), \|\cdot\|_K)$ satisfies for all $x \in X$ and all $w \in \mathbb{R}^m$,

$$J_u^*(w)(x) = \frac{1}{m} \sum_{i=1}^m w_i K(x, u_i).$$

Proof. (i) Continuity of J_u follows from the definition of a RKHS.
 (ii) Every linear subspace of a finite dimensional space is closed and so is $R(J_u)$.
 (iii) As every continuous operator with a finite range is compact [22, p. 188], so is J_u .
 (iv) By the reproducing property (9) and the definition of an adjoint operator, for every $x \in X$ and every $w \in \mathbb{R}^m$ we have $J_u^*(w)(x) = \langle J_u^*(w), K_x \rangle_K = \langle w, J_u(K_x) \rangle_{2,m} = \frac{1}{m} \sum_{i=1}^m w_i K(x, u_i)$. □

We denote by

$$\mathcal{E}_{z,\gamma,K} = \mathcal{E}_z + \gamma \|\cdot\|_K^2$$

the Tikhonov regularization of the empirical error \mathcal{E}_z with the stabilizer $\|\cdot\|_K^2$ and the regularization parameter γ . The role of kernel norms as stabilizers in Tikhonov's regularization (5) can be intuitively well understood in the case of convolution kernels, i.e., kernels $K(x, y) = k(x - y)$ defined as translations of a function $k \in \mathcal{L}^1(\mathbb{R}^d) \cap \mathcal{L}^2(\mathbb{R}^d)$, for which the Fourier transform \hat{k} is positive. For such kernels, the value of the stabilizer $\|\cdot\|_K^2$ at any $f \in \mathcal{H}_K(\mathbb{R}^d)$ can be expressed as

$$\|f\|_K^2 = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} \frac{\hat{f}(\omega)^2}{\hat{k}(\omega)} d\omega \tag{10}$$

[827]. So when $\lim_{\|\omega\| \rightarrow \infty} 1/\hat{k}(\omega) = \infty$, the stabilizer $\|\cdot\|_K^2$ plays a role of a high-frequency filter. An example of a convolution kernel with a positive Fourier transform is the Gaussian kernel.

The next theorem describes argminima of empirical error \mathcal{E}_z and its regularized modification $\mathcal{E}_{z,K,\gamma}$. For a kernel $K : X \times X \rightarrow \mathbb{R}$ and a vector $u \in X^m$, $\mathcal{K}[u]$ denotes the Gram matrix of the kernel K with respect to the vector u , i.e., the matrix

$$\mathcal{K}[u]_{i,j} = K(u_i, u_j),$$

$\mathcal{K}_m[x]$ denotes the matrix $\frac{1}{m}\mathcal{K}[u]$, and \mathcal{I}_m the identity $m \times m$ matrix.

Theorem 1. Let $X \subseteq \mathbb{R}^d$, $K : X \times X \rightarrow \mathbb{R}$ be a symmetric positive semidefinite kernel, m be a positive integer, $z = (u, v)$ with $u = (u_1, \dots, u_m) \in X^m$, $v = (v_1, \dots, v_m) \in \mathbb{R}^m$, then

(i) there exists an argminimum f^+ of \mathcal{E}_z over $\mathcal{H}_K(X)$, which satisfies

$$f^+ = J_u^+(v) = \sum_{i=1}^m c_i K_{u_i}, \tag{11}$$

where

$$c = (c_1, \dots, c_m) = \mathcal{K}[u]^+ v,$$

and for all $f^o \in \text{argmin}(\mathcal{H}_K(X), \mathcal{E}_z)$, $\|f^+\|_K \leq \|f^o\|_K$;

(ii) for all $\gamma > 0$, there exists a unique argminimum f^γ of $\mathcal{E}_{z,\gamma,K}$ over $\mathcal{H}_K(X)$, which satisfies

$$f^\gamma = J_u^\gamma(v) = \sum_{i=1}^m c_i^\gamma K_{u_i}, \tag{12}$$

where

$$c^\gamma = (c_1^\gamma, \dots, c_m^\gamma) = (\mathcal{K}_m[u] + \gamma \mathcal{I}_m)^{-1} v;$$

(iii) $\lim_{\gamma \rightarrow 0} \|f^\gamma - f^+\|_K = 0$.

Proof. (i) By the representation (4), argminimum of \mathcal{E}_z over $\mathcal{H}_K(X)$ is a pseudosolution of an inverse problem given by the operator J_u for the data v . By Proposition 3 (i) and (ii), J_u is continuous and has a closed range, thus by (6), we obtain $J_u^+ = J_u^*(J_u J_u^*)^+$. Proposition 3(iii) implies that $J_u J_u^* : \mathbb{R}^m \rightarrow \mathbb{R}^m$ can be expressed by the matrix $\mathcal{K}_m[u]$. So $f^+ = J_u^+(v) = \sum_{i=1}^m c_i K_{u_i}$, where $c = \mathcal{K}_m[u]^+ v$.

- (ii) By (7), $f^\gamma = J_u^\gamma(v) = J_u^*(J_u J_u^* + \gamma I_m)^{-1}v$, where I_m denotes the identity operator on \mathbb{R}^m . Thus $f^\gamma = \sum_{i=1}^m c_i^\gamma K_{u_i}$, where $c^\gamma = (\mathcal{K}_m[u] + \gamma \mathcal{I}_m)^{-1}v$.
- (iii) By (8), $\lim_{\gamma \rightarrow 0} \|f^\gamma - f^+\|_K = \lim_{\gamma \rightarrow 0} \|J_u^\gamma(v) - J_u^+(v)\|_K = 0$ □

Theorem 1(ii) became well-known under the name ‘‘Representer Theorem’’ (15). For X compact and K continuous, several authors derived it using Fréchet derivatives (see, e.g., (11), (15), (28)). Our proof of Theorem 1 based on theory of inverse problems needs neither compactness nor continuity and it also includes non regularized case. Comparison of regularized and non regularized case shows that regularization merely modifies coefficients of the linear combination of functions composing the argminimum.

Theorem 1 shows that for every symmetric positive semidefinite kernel K and every sample of empirical data z , the argminimum f^+ of the empirical error functional \mathcal{E}_z over the whole space $\mathcal{H}_K(X)$ is formed by a linear combination of the representer K_{u_1}, \dots, K_{u_m} of the input data u_1, \dots, u_m . The pseudosolution f^+ can be interpreted as an *input-output function of a network with one hidden layer with kernel units and a single linear output unit*. The coefficients $c = (c_1, \dots, c_m)$ of the linear combination (corresponding to the output weights of the network) satisfy $c = \mathcal{K}_m[u]^+v$, so the output weights can be obtained by solving the system of linear equations.

As the operator J_u has finite dimensional range, it is compact and thus its pseudoinverse J_u^+ is unbounded. So the optimal solution f^+ of minimization of the empirical error \mathcal{E}_z is unstable with respect to a change of output data v . Stability can be improved by replacing the pseudosolution $f^+ = J_u^+(v)$ with the regularized solution $f^\gamma = J_u^\gamma(v)$, which is a linear combination of the same functions K_{u_1}, \dots, K_{u_m} . But the coefficients of these two linear combinations are different: in the regularized case $c^\gamma = (\mathcal{K}_m[u] + \gamma \mathcal{I})^{-1}v$, while in the non-regularized one $c = \mathcal{K}_m[u]^+v$.

For a convolution kernel $K(x, y) = k(x - y)$, all functions of the form $f = \sum_{i=1}^m w_i K_{u_i}$, which are computable by one-hidden layer networks with kernel units computing translations of k , satisfy

$$\|f\|_K \leq \sum_{i=1}^m |w_i| \|K_{u_i}\|_K = \sum_{i=1}^m |w_i| K(u_i, u_i) = \sum_{i=1}^m |w_i| k(0) = \sum_{i=1}^m |w_i|.$$

So

$$\|f\|_K^2 \leq \left(\sum_{i=1}^m |w_i| \right)^2. \tag{13}$$

In practical learning tasks, an output-weight regularization which penalizes input-output functions with large ℓ_1 or ℓ_2 -norms of output-weight vectors has been widely used for its simplicity (see, e.g., (1)). The inequality (13) shows that an output-weight regularization also penalizes solutions with large $\|\cdot\|_K$ -norms.

In typical applications, networks with much smaller number n of units than the size of the training sample of data m are used. However, characterization of theoretically optimal solutions achievable over networks with the numbers of units equal to the sizes m of training data can be used to estimate speed of convergence of suboptimal solutions of the optimization task to the optimal ones. Some estimates of this speed were derived in (29,30).

5 Minimization of Empirical Error over Spaces of Continuous Functions

A continuous kernel $K : X \times X$ with $X \subset \mathbb{R}^d$ compact is called *universal* if $\mathcal{H}_K(X)$ is dense in the space of continuous functions with the supremum norm $(\mathcal{C}(X), \|\cdot\|_{\text{sup}})$ [31]. By Propositions 1 and 2 all argminima of the empirical error \mathcal{E}_z over spaces $\mathcal{H}_K(X)$ with universal kernels K are also argminima over the whole space $\mathcal{C}(X)$.

The concept of universal kernel is related to *universal approximation property* of a class of networks. A one-hidden layer network with units from a dictionary $G_\phi(A)$ has a universal approximation property in $(\mathcal{C}(X), \|\cdot\|_{\text{sup}})$ if $\text{span } G_\phi(A)$ is dense in $(\mathcal{C}(X), \|\cdot\|_{\text{sup}})$. As the set of representers $G_K(X) = \{K_x | x \in X\}$ is a subset of $\mathcal{H}_K(X)$, universal approximation property of a class of networks with units from the dictionary $G_K(X)$ implies that K is a universal kernel.

The next theorem by Mhaskar [32] states that Gaussian networks with any fixed width are universal approximators. Let $S_d^b : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ denotes the d -dimensional Gaussian kernel i.e.,

$$S_d^b(x, y) = e^{-b\|x-y\|^2},$$

and $S_d^b[u]$ denote the Gramm matrix of the kernel S_d^b with respect to the vector u .

Theorem 2. *Let d be a positive integer and $X \subset \mathbb{R}^d$ be compact. Then for all $b > 0$, $\text{span } S_d^b(X)$ is dense subspaces of $(\mathcal{C}(X), \|\cdot\|_{\text{sup}})$.*

As $\text{span } S_d^b(X)$ is dense in and $\mathcal{H}_{S_d^b}(X)$, also $\mathcal{H}_{S_d^b}(X)$ is dense in $(\mathcal{C}(X), \|\cdot\|_{\text{sup}})$. Note that proof of this theorem employs properties of derivatives of Gaussians (which are products of Hermite polynomials with the Gaussian) and thus it cannot be extended to other kernels. Examples of universal kernels are exponential kernel $K(x, y) = e^{x \cdot y}$ on all compact $X \subset \mathbb{R}^d$ and binomial kernel $K(x, y) = (1 - x \cdot y)^{-\alpha}$ on the unit disk in \mathbb{R}^d [31].

The next theorem describes a set of argminima of empirical error over the space $\mathcal{C}(X)$. For a set A , $\text{conv } A$ denotes the *convex hull* of A .

Corollary 1. *Let X be a compact subset of \mathbb{R}^d , $z = (u, v)$, where $u = (u_1, \dots, u_m) \in X^m$, $v = (v_1, \dots, v_m) \in \mathbb{R}^m$. Then the set of argminima of \mathcal{E}_z over $\mathcal{C}(X)$ contains the set $\text{conv}\{f_b^+ | b > 0\}$, where $f_b^+ = \sum_{i=1}^m c_i^b S_{d u_i}^b$ with $c^b = S_d^b[u]^+ v$.*

Proof. By Theorem 1(i) for every $b > 0$, $f_b^+ = \sum_{i=1}^m c_i^b S_{d u_i}^b$ is an argminimum of \mathcal{E}_z over $\mathcal{H}_{S_d^b}(X)$. By Theorem 2 (i), $\text{span } S_d^b(X)$ is dense in $(\mathcal{C}(X), \|\cdot\|_{\text{sup}})$ and by Proposition 1 \mathcal{E}_z is continuous on $(\mathcal{C}(X), \|\cdot\|_{\text{sup}})$. As $f_b^+ \in \text{span } S_d^b(X)$, by Proposition 2 f_b^+ is an argminimum of \mathcal{E}_z over $\mathcal{C}(X)$. As the set of argminima is convex, the statement holds. \square

Corollary 1 shows that in the space of continuous functions $\mathcal{C}(X)$, for each width $b > 0$ of the Gaussian kernel, there exists an argminimum of \mathcal{E}_z formed by a linear combination of Gaussians with the width b . All these Gaussians have the same centers given by the input data. It was proven in [33] that the set $\{e^{-b\|\cdot-y\|} | b \in \mathbb{R}_+, y \in \mathbb{R}^d\}$ of all Gaussians with varying widths and centers is linearly independent. So the set of all

these argminima of \mathcal{E}_z is linearly independent and all their convex combinations are also argminima. As Gaussians are known to interpolate the data exactly [4], the empirical error achieves the value zero at a large convex set.

The next theorem describes a relationship of kernel norms for Gaussian kernels with different widths.

Corollary 2. For all $b > 0$, $\mathcal{H}_{S_a^b}(\mathbb{R}^d) = \{f \in \mathcal{L}^2(\mathbb{R}^d) \mid \|f\|_{S_a^b} < \infty\}$, where

$$\|f\|_{S_a^b}^2 = \left(\frac{b}{\sqrt{\pi}}\right)^d \int_{\mathbb{R}^d} \frac{\hat{f}(s)^2}{e^{-\|s\|_{\frac{a}{2b}}^2}} ds$$

and for all $b \geq a > 0$

$$\|f\|_{S_a^b} \leq \left(\frac{b}{a}\right)^{d/2} \|f\|_{S_a^a}.$$

Proof. By (10) and the formula

$$\widehat{e^{-a\|\cdot\|^2}}(s) = (\sqrt{2a})^{-d} e^{-\|s\|_{\frac{a}{2a}}^2}$$

one gets

$$\frac{\|f\|_{S_a^b}^2}{\|f\|_{S_a^a}^2} = \left(\frac{b}{a}\right)^d \frac{\int_{\mathbb{R}^d} \hat{f}(s)^2 e^{\|s\|_{\frac{a}{2b}}^2} ds}{\int_{\mathbb{R}^d} \hat{f}(s)^2 e^{\|s\|_{\frac{a}{2a}}^2} ds}.$$

As $a \leq b$ implies $e^{\|s\|_{\frac{a}{2b}}^2} \leq e^{\|s\|_{\frac{a}{2a}}^2}$, we have $\frac{\|f\|_{S_a^b}}{\|f\|_{S_a^a}} \leq \left(\frac{b}{a}\right)^{d/2}$. □

So Corollary 2 shows that with sharpening the Gaussian, the size of the norm on the induced RKHS is decreasing exponentially fast. The ratio of sizes of stabilizers in the form of squares of norms on Gaussian RKHSs with width a and b such that $a < b$, grows with increasing dimension exponentially as $\left(\frac{b}{a}\right)^d$. Thus for high dimensions d , choice of a width of the Gaussian kernel strongly influences regularization.

Acknowledgements. This work was partially supported by by GA ČR grant P202/11/1368 and the Institutional Research Plan AV0Z10300504.

References

1. Fine, T.L.: Feedforward Neural Network Methodology. Springer, Heidelberg (1999)
2. Kecman, V.: Learning and Soft Computing. MIT Press, Cambridge (2001)
3. Ito, Y.: Finite mapping by neural networks and truth functions. Mathematical Scientist 17, 69–77 (1992)
4. Michelli, C.A.: Interpolation of scattered data: Distance matrices and conditionally positive definite functions. Constructive Approximation 2, 11–22 (1986)
5. Girosi, F., Poggio, T.: Regularization algorithms for learning that are equivalent to multilayer networks. Science 247(4945), 978–982 (1990)

6. Girosi, F., Jones, M., Poggio, T.: Regularization theory and neural networks architectures. *Neural Computation* 7, 219–269 (1995)
7. Bishop, C.: Training with noise is equivalent to Tikhonov regularization. *Neural Computation* 7(1), 108–116 (1995)
8. Girosi, F.: An equivalence between sparse approximation and support vector machines. *Neural Computation* 10, 1455–1480 (1998)
9. Aronszajn, N.: Theory of reproducing kernels. *Transactions of AMS* 68, 337–404 (1950)
10. Parzen, E.: An approach to time series analysis. *Annals of Math. Statistics* 32, 951–989 (1966)
11. Wahba, G.: *Splines Models for Observational Data*. SIAM, Philadelphia (1990)
12. Boser, B.E., Guyon, I.M., Vapnik, V.: A training algorithms for optimal margin classifiers. In: Haussler, D. (ed.) *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152. ACM Press, Pittsburg (1992)
13. Cortes, C., Vapnik, V.: Support vector networks. *Machine Learning* 20, 273–297 (1995)
14. Schölkopf, B., Smola, A.J.: *Learning with Kernels – Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, Cambridge (2002)
15. Cucker, F., Smale, S.: On the mathematical foundations of learning. *Bulletin of AMS* 39, 1–49 (2002)
16. Hansen, P.C.: *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, Philadelphia (1998)
17. Bertero, M.: Linear inverse and ill-posed problems. *Advances in Electronics and Electron Physics* 75, 1–120 (1989)
18. Engl, E.W., Hanke, M., Neubauer, A.: *Regularization of Inverse Problems*. Kluwer, Dordrecht (1999)
19. Kůrková, V.: Learning from data as an inverse problem. In: Antoch, J. (ed.) *COMPSTAT 2004 - Proceedings on Computational Statistics*, pp. 1377–1384. Physica-Verlag/Springer, Heidelberg (2004)
20. Kůrková, V.: Neural network learning as an inverse problem. *Logic Journal of IGPL* 13, 551–559 (2005)
21. Vito, E.D., Rosasco, L., Caponnetto, A., Giovannini, U.D., Odone, F.: Learning from examples as an inverse problem. *Journal of Machine Learning Research* 6, 883–904 (2005)
22. Friedman, A.: *Modern Analysis*. Dover, New York (1982)
23. Tikhonov, A.N., Arsenin, V.Y.: *Solutions of Ill-posed Problems*. W.H. Winston, Washington (1977)
24. Moore, E.H.: Abstract. *Bulletin of AMS* 26, 394–395 (1920)
25. Penrose, R.: A generalized inverse for matrices. *Proceedings of Cambridge Philosophical Society* 51, 406–413 (1955)
26. Groetch, C.W.: *Generalized Inverses of Linear Operators*. Dekker, New York (1977)
27. Loustau, S.: Aggregation of SVM classifiers using Sobolev spaces. *Journal of Machine Learning Research* 9, 1559–1582 (2008)
28. Poggio, T., Smale, S.: The mathematics of learning: dealing with data. *Notices of AMS* 50, 537–544 (2003)
29. Kůrková, V., Sanguineti, M.: Error estimates for approximate optimization by the extended Ritz method. *SIAM Journal on Optimization* 15, 461–487 (2005)
30. Kůrková, V., Sanguineti, M.: Learning with generalization capability by kernel methods with bounded complexity. *Journal of Complexity* 13, 551–559 (2005)
31. Steinwart, I., Christmann, A.: *Support Vector Machines*. Springer, New York (2008)
32. Mhaskar, H.N.: Versatile Gaussian networks. In: *Proceedings of IEEE Workshop of Nonlinear Image Processing*, pp. 70–73 (1995)
33. Kůrková, V., Neruda, R.: Uniqueness of functional representations by Gaussian basis function networks. In: *Proceedings of ICANN 1994*, pp. 471–474. Springer, London (1994)

A Cortically Inspired Learning Model

Atif Hashmi and Mikko Lipasti

Department of Electrical and Computer Engineering, University of Wisconsin
1415 Engineering Drive, Madison, WI - 53706, U.S.A.
ahashmi@wisc.edu, mikko@engr.wisc.edu
<http://www.ece.wisc.edu/~pharm>

Abstract. We describe a biologically plausible learning model inspired by the structural and functional properties of the cortical columns present in the mammalian neocortex. The strength and robustness of our model is ascribed to its biologically plausible, uniformly structured, and hierarchically distributed processing units with their localized learning rules. By modeling cortical columns rather than individual neurons as our fundamental processing units, we get hierarchical learning networks that are computationally less demanding and better suited for studying higher cortical properties like independent feature detection, plasticity, etc. Another interesting attribute of our model is the use of feedback processing paths to generate invariant representation to robustly recognize variations of the same patterns and to determine the set of features sufficient for recognizing different patterns in the input dataset. We train and test our hierarchical networks using synthetic digit images as well as a subset of handwritten digit images obtained from the MNIST database. Our results show that our cortical networks use unsupervised feedforward processing as well as supervised feedback processing to robustly recognize handwritten digits.

Keywords: Cortical algorithms, Cortical columns, Invariant representation, Feedforward and feedback processing, Pruning, Automatic abstraction, Fault tolerance.

1 Introduction

Understanding of the structural and operational aspects of various components of the mammalian neocortex has significantly increased over the past few decades [1,2,3,4]. This has led to the development of both low level biologically realistic as well as high level biologically inspired computational models. Low level biologically realistic models include the blue brain project [5], DARPA's SyNAPSE project [6], etc. These models use neurons as their basic implementation abstraction and simulate detailed low level behavior of these neurons. Most of these models use Hebbian learning rules [7,8] along with Spike Timing Dependent Plasticity (STDP) [9] for learning and information processing. This makes them quite complex and computationally very expensive. To cope with these issues, high level learning models inspired by the neocortical properties have been proposed. These models implement various neocortical attributes like uniform structure, hierarchy, spatial pooling, temporal pooling, etc. Some of these models include ART [10], HTM [11], Bayesian networks [12], and deep belief networks [13].

Even though these models are computationally quite efficient and implement some behavioral aspects of the neocortex, they are quite divorced from the actual biological structure and properties of the neocortex.

We hypothesize that to develop intelligent models as powerful as the brain, we must adhere to structural and functional properties of the biological example. In this article, we describe a cortical model that uses cortical columns, found in the mammalian neocortex [14], as its basic structural and functional processing units. Since cortical columns are our basic implementation abstraction, our model is inherently computationally efficient and is biologically plausible as well. Our model uses unsupervised feedforward processing and plasticity principles to learn and extract independent features from the input patterns and it uses supervised feedback processing, object permanence, and temporal associativity to develop invariant representations for variations of the same pattern. Using the feedback from higher levels, our model is also able to determine the set of independent features that are sufficient to recognize the different patterns. This is in accordance with the biological studies which show that the human brain relies on a small subset of features in the visual scene to recognize objects [15][16].

To test and validate our cortical model, we use synthetic digit images as well as a subset of handwritten digit images obtained from the MNIST database [17]. Our results show that our cortical networks learn to identify each of the unique digits present in the sample set and also pools variations of the same digit together to develop invariant representations. Moreover, using the feedback from higher levels, our model is also able to determine the set of features that is sufficient to differentiate between digits.

2 Cortical Structures, Organization, and Processing

The human brain can be divided into two main parts: the old brain and the new brain. The old brain mainly constitutes those parts of brain that developed early in evolution. These include pathways from sensory modalities to the new brain, spinal cord, and other parts that deal with instinctual behavior. The new brain, also referred to as the *neocortex*, is part of the brain which is unique to mammals and is highly developed for humans; it accounts for about 77% of the human brain (in volume) [18]. The neocortex is responsible for perception, language, imagination, mathematics, arts, music, planning, and all the other aspects necessary for an intelligent system. It contains virtually all our memories, knowledge, skills, and experiences.

A very intriguing property of the neocortex is its apparent *structural and functional uniformity* [14][19], i.e. the regions of the neocortex that process auditory inputs, appear very similar to the regions that handle visual and other inputs. This uniformity suggests that even though different regions specialize in different tasks, they employ the same underlying algorithm. In essence, the neocortex is a hierarchy of millions of seemingly-identical functional units that are called *cortical columns*. The concept of cortical columns was introduced by Mountcastle in his seminal paper in 1957 [20]. Since then, this concept has been widely accepted and studied. Later studies showed that cortical columns could further be classified into *minicolumns* and *hypercolumns* [21][4][3]. A hypercolumn contains about 50 to 100 minicolumns, and each of these minicolumns consists of around 200 to 500 neurons. The term cortical column is

sometimes used for both types of columns, though, in literature, it usually refers to hypercolumns. The minicolumns within the same hypercolumn share the same receptive field (set of input connections) and are strongly connected with each other via *inhibitory lateral connections*. Studies [21,22] hypothesize that the minicolumns use these inhibitory paths to learn unique/independent features from the input patterns. These hypercolumns are then arranged in the form of a hierarchy throughout the neocortex. Information flows up this hierarchy via *excitatory feedforward paths* and flows down the hierarchy through *feedback paths*.

It is believed that cortical regions operate by progressively abstracting and manipulating increasingly complex notions throughout the neural hierarchy [23]. For instance, from a visual scene, the visual cortex first identifies segments of different orientations, then elementary shapes such as angles and intersections, and increasingly complex combinations, such as objects found in our environment [24]. This automatic abstraction capability for various inputs (visual, auditory, olfactory) partly explains why the neocortex still outperforms traditional computers on a number of tasks, such as object recognition, language learning, and motor control. Emulating such capability is thus a major step in building computing systems that can compete with the processing characteristics of the brain.

Although there exists a monumental amount of literature explaining the aforementioned properties of the neocortex, certain alternative properties of the neocortex have not been sufficiently explored. One of these properties is the ability to determine the set of features that the neocortex uses to recognize objects. Obviously, the neocortex does not require all of the object's features to recognize it. For example, Maw et.al [15] show that while recognizing a human face, subjects tend to gaze more at the eyes, lips, and nose. Thus, out of all the features that constitute a human face the neocortex uses a small subset to process the facial recognition task. Sigala et.al [16] shows similar results in more details. Thus, within the neocortex, there exists of notion of a sets of features that are sufficient to differentiate visual patterns.

3 Cortical Model Description

3.1 Hypercolumn Abstraction

As mentioned in Section 1, we model cortical columns as our basic structural and functional implementation abstraction. Figure 1 shows the architecture of the basic functional unit in our cortical model. A hypercolumn consists of multiple minicolumns that are strongly connected with each other via horizontal inhibitory connections. All of the minicolumns within a hypercolumn share the same receptive field. A receptive field is defined as the region within sensory input that is associated to a hypercolumn.

3.2 Unsupervised Feedforward Processing and Independent Feature Learning

In our model each of the minicolumns within a hypercolumn learns to identify independent features from the input patterns using lateral inhibitory paths. This is in accordance with the biological discussion presented in Section 2. In this section, we provide detailed discussion on how each of the minicolumns learns to identify unique patterns without any supervision.

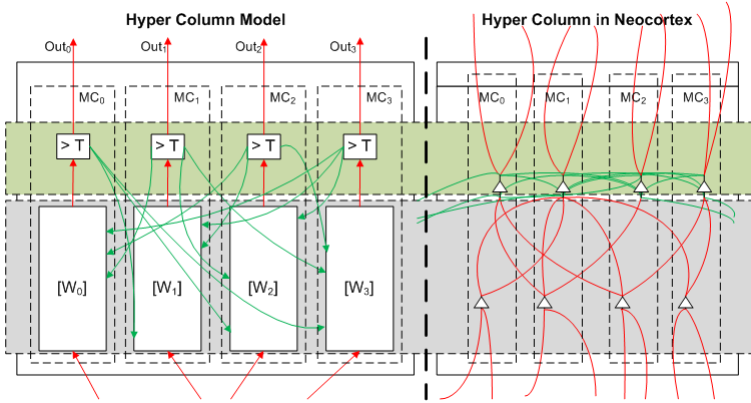


Fig. 1. Mapping between hypercolumn network and feedforward circuitry of a hypercolumn in the neocortex. Left: A hypercolumn network in our model with four minicolumns. Right: Structure of a biological hypercolumn.

Random Activations and Initial Learning. Initially all the minicolumns within a hypercolumn are initialized with very weak random weights. Thus, they show no preference for any pattern that might occur within their receptive field. Since our minicolumns also model the stochastic nature of neurons by including random neocortical firing behavior [25][26], they exhibit high activations over random intervals. When the random activation of a specific minicolumn coincides frequently with various stable occurrences of the same pattern, the minicolumn adjusts its weights so that the correlation between the weights and the input patterns increases. Thus over time, that minicolumn develops a firing preference for that specific pattern. While this random activation of minicolumns may not initially seem productive, this behavior is harnessed to make the model fault-tolerant, improves the model’s training time, and mimics the behavior of its biological inspirations.

Evaluating Output of Minicolumns. Each of the minicolumns contains a set of weights W initialized to random values which are close to zero. During each training epoch, each of the minicolumns evaluates the dot-product $DP = \sum_{i=1}^N X_i \cdot W_i$ between its weights W and the input X . The result of the dot-product becomes the input to the activation function given by,

$$\frac{1.0}{1.0 + e^{(-\frac{DP - cutoff}{\beta})}} + \alpha \times \sum |W_i| \tag{1}$$

Here, $cutoff = \phi \times \sum |W_i|$. ϕ determines the error tolerance of the minicolumn. β defines the sharpness of the activation function while α controls the effect of weight strength of a minicolumn on its output. The minicolumn is said to fire if the value of its activation function is greater than a determined threshold.

Lateral Inhibition and Independent Feature Identification. When an input X is presented to the hypercolumn, none of the untrained minicolumns fire for that input.

However, if the random firing activity of a minicolumn coincides with the occurrence of an input pattern, that minicolumn adjusts its weights so that the dot-product between the input and the weights is improved. This is achieved by strengthening the weights corresponding to the inputs X_i that are currently active. Thus, over multiple iterations a minicolumn learns to identify a feature that initially coincided with the random activity of the minicolumn. At the same time, each minicolumn inhibits neighboring minicolumns from firing via lateral inhibitory connections for the pattern it has already learned to recognize. If multiple minicolumns fire at the same time, the one with the strongest response inhibits the ones with weaker responses. The inhibited minicolumns then weaken their weights corresponding to highly active X_i so that their dot-product with the input is minimized. As a result of this process, the hypercolumn network is able to recognize unique patterns without any supervision. A very interesting byproduct of having minicolumns learn independent features through lateral inhibition is inherent fault tolerance i.e. if a minicolumn that was firing for a feature suddenly dies (permanent hardware or software error in a future synthetic application), over time, another available neighboring minicolumn will start firing for that feature. This makes our hypercolumn structure inherently tolerant to permanent faults.

Weight Update Rules. Each time a minicolumn fires it modifies its weights so that its correlation with the input pattern that has caused it to fire increases. Weights are strengthened using the following update rule.

$$W_i = X_i \times \left(W_i + \left(C_1 + \gamma \times \frac{1.0}{1.0 + e^{\left(-\frac{W_i - C_2}{\beta}\right)}} \right) \right) \quad (2)$$

Here, X_i is the input corresponding to W_i , C_1 defines the minimum amount of update added to the current W_i and C_2 defines how the present W_i will affect the weight update. In our weight strengthening rule, the update added to W_i is dependent upon the present value of W_i as well. This means that if W_i is strong it will get a higher update value. This is in accordance with biological data [26,27].

In the case when a minicolumn is inhibited, it modifies the weights using the following update rule.

$$W_i = X_i \times (W_i - \delta) \quad (3)$$

Here, δ defines the weight update rate in the presence of inhibition.

3.3 Hierarchical Arrangement of Hypercolumns

To perform complex tasks the hypercolumns can be arranged in the form of a hierarchy. Lower hierarchical levels identify simple features and communicate their output to the higher levels via feedforward paths. Each of the higher level hypercolumns receives inputs from multiple lower level hypercolumns. In this manner the activations flow up the hierarchy and the minicolumns in the top-level hypercolumns train themselves to identify each of the complex unique pattern from the input. Each level of this hierarchy behaves the same way as different levels of the visual cortex i.e. lower level hypercolumns detect edges, and the hypercolumns at the higher levels detect progressively complex features. It should be noted that our hierarchical model supports any complex hierarchical arrangement of hypercolumns.

3.4 Supervised Feedback Processing and Invariant Representations

Our feedforward learning process enables our cortical hierarchy to learn unique features from the input patterns. Each of the minicolumns can withstand and fire for patterns with small variations but patterns with significant variations are recognized as different features. This means that two variations of the same pattern might be recognized as two different features. To resolve this issue and generate invariant representation for variations of the same pattern, we make use of our supervised feedback processing algorithm.

Algorithm 1. Pseudo code for generating invariant representations within a minicolumn using supervised feedback

```

if feedback > 0 then
  if hasNotFired then
    if hasMaxFiringHistory then
      UpdateSynapticWtsExcitatory(feedback)
    end if
  end if
else
  if hasMaxFiringHistory then
    UpdateSynapticWtsExcitatory(feedback)
  if isStable then
    for i = 1 to N do
      if IsActive(child[i]) then
        SendFBToChild(i, feedback)
      end if
    end for
  end if
else
  UpdateSynapticWtsInhibitory(feedback)
end if
end if
end if

```

Lets assume that our hierarchical network has started to recognize a pattern. Now it is exposed to another variation of the same patterns that is quite different from the previous one e.g. two different variations of a handwritten digit. At this point, only some of the minicolumns within the hierarchy might fire. As a result, the top level minicolumn that is supposed to fire for that pattern might not fire. If this behavior persists, new minicolumns will train themselves to recognize features in the new variation that are quite different from the original pattern. Over time, that new variation will be identified as a new pattern. This will be marked by firing of a minicolumn in the top level of the hierarchy. At this point, the top level hypercolumn receives a feedback signal. This feedback signal forces the minicolumn firing for the original pattern to fire and also inhibits the minicolumn that is firing for the new variation. Now, the minicolumn receiving excitatory feedback also adjusts its weights so that it fires for the new variation as well while the inhibited minicolumn changes its weights so that it does not fire for that input pattern. Thus over multiple exposures, the minicolumn firing for the original pattern will also start to fire for the new variation. Once the top level minicolumn starts to give a

stable activation for both the variations, it will start to send the feedback signal down so that lower level minicolumns can also create invariant representations. The amount of feedback sent to each of the lower level minicolumns is proportional to its firing history i.e. if a minicolumn has been firing a lot in the past, it will get stronger feedback. Thus, over time most active minicolumn ends up pooling its child minicolumns to generate invariant representations and inhibits its neighbors from firing. This results in significant resource optimization. The process of generating invariant representations within a minicolumn using feedback is explained in the pseudo-code provided in Algorithm 1. In Algorithm 1, *UpdateSynapticWtsExcitatory* models the functionality of Equation 2 while *UpdateSynapticWtsInhibitory* models Equation 3.

3.5 Learning Spatial Correlations from Past Experiences

To determine the set of features sufficient for recognition of unique images, we consider the spatial correlations that exist among the occurrence of different independent features constructing the objects in the dataset. Since each of the minicolumns trains itself to identify independent features from its input, we can determine the spatial correlation among various minicolumns by observing their firing patterns.

Table 1. Location of minicolumns in the hypercolumn hierarchy identifying different features and shapes

Level	Hypercolumn	Minicolumn	Recognizes
0	0	0	Feature a
0	0	1	Feature b
0	0	2	Feature c
0	1	0	Feature a
0	1	1	Feature b
0	1	2	Feature c
1	0	0	Shape A
1	0	1	Shape B
1	0	2	Shape C
1	0	3	Shape D
1	0	4	Shape E

To illustrate our methodology for determining the spatial correlations, we use a simple 2-level hierarchical network shown in Figure 2. Assume that this network is trained using the dataset shown in Figure 3. Each of the four elements of the dataset is a synthetic image of size 2x4. Each image can be divided into two halves, i.e. the first 2x2 grid and the second 2x2 grid separated by the dotted line in Figure 3. Within each of the two grids, only three unique features occur: a horizontal line across the top (Feature a), a vertical line on the left (Feature b), and a diagonal line (Feature c). Once the network is trained with each of the 2x4 images, each of the minicolumns within the Level 0 hypercolumns start to recognize any one of the three unique features. At the same time, each of the four minicolumns at Level 1 starts to recognize any one of the four

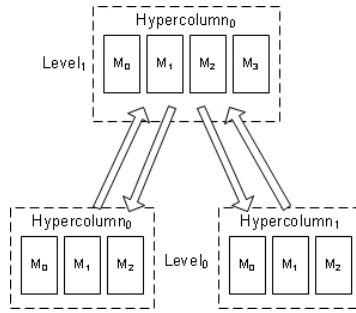


Fig. 2. Block level diagram of a 2-level hierarchical hypercolumn network

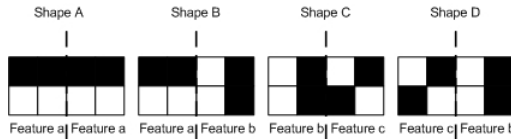


Fig. 3. Set of features used to train 2-level hierarchical network

different shapes. Table 1 shows the details about the patterns identified by each of the minicolumns in different hypercolumns in the 2-level hierarchy.

Once the 2-Level hierarchy is in a stable state i.e. all the four shapes are recognized by the Level 1 hypercolumn, the Level 1 hypercolumn starts to create spatial correlations among the occurrences of different features recognized by the minicolumns in each of the Level 0 hypercolumns. These spatial correlations help in determining the set of features sufficient for recognition of each of the unique shapes. A spatial correlation is created between two features if they co-occur frequently. Once a spatial correlation has been created, occurrence of a feature can be predicted by any occurrence of the feature it is spatially correlated to. For example, in Figure 3 feature *a* in the first 2x2 grid always occurs with features *a* and *b* in the second 2x2 grid. This means that a spatial correlation between feature *a* on the left 2x2 grid and features *a* and *b* on the right 2x2 grid is developed. Feature *b* on the left 2x2 grid co-occurs with features *a* and *c* on the right 2x2 grid. Thus, a spatial correlation is maintained among these features. Finally, feature *c* on the left 2x2 grid only co-occurs with feature *b* on the right 2x2 grid so there is just one spatial correlation for feature *c* i.e. feature *b*.

Figure 4 shows the associations among various features that are created due to the spatial correlations that exist among them. In the figure, we can see that Level 0 Hypercolumn 0 Minicolumn 0 (L0H0M0) is connected to L0H1M0 through L1H0M0 which detects Shape A. Similarly, L0H0M0 is connected to L0H1M1 through L1H0M1 which detects Shape B. Other associations in Figure 4 are also created in the same manner. Figure 4 details an association graph that shows the spatial correlations which exist among different features that are detected by the Level 0 minicolumns.

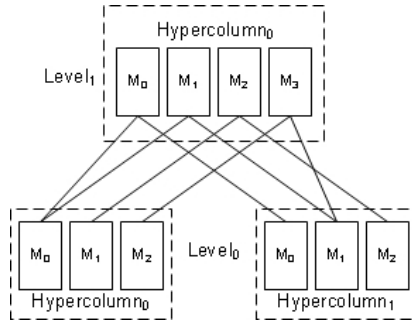


Fig. 4. A graphical representation of spatial correlations created among various features recognized by the Level 0 minicolumns after the hierarchical network reaches a steady state

4 Determining the Set of Features Sufficient for Recognition of Unique Shapes

To determine the set of features sufficient for identifying the shapes exposed to the network, we use the spatial correlation graph like the one shown in Figure 4. At each level of the hierarchy, the spatial correlations between different minicolumns are observed and the associations that provide redundant information are pruned. For example, if minicolumn A is just spatially associated to minicolumn B and minicolumn B is spatially associated to some other minicolumn a well, then the feature recognized by minicolumn A is sufficient enough to identify the shape typically recognized by minicolumns A and B together. Thus, the association from minicolumn B can be pruned. Finally, if a minicolumn has no associations coming out of it, then that minicolumn can also be pruned.

To better explain our algorithm, we apply it to the spatial correlation graph shown in Figure 4. Initially, L0H0M0 is selected. Since there are two edges out of L0H0M0 it is left as is. Then L0H0M1 is selected. Since there is just one association from L0H0M1, the association between L0H1M2 and L1H0M2 is declared useless and is pruned. For L0H0M2, there is again just one edge connecting L0H0M2 to L0H1M1 through L1H0M3. Thus, the edge from L0H1M1 to L1H0M3 is pruned because L1H0M3 can recognize Shape 4 just by using the information provided by L0H0M2. The same is true for L0H1M0. Thus, the edge between L0H0M0 and L1H0M0 is pruned. Note that in many cases the pruning algorithm has the freedom to choose which redundant connections to retain, and which to prune. After this no more changes to the network connectivity take place because all of the remaining edges provide useful information to recognize each of the unique shapes. The resultant graph after applying the spatial correlation based algorithm is shown in Figure 5.

In Figure 5, we see that after applying our algorithm, five out of ten edges between Level 0 and Level 1 are pruned. Furthermore, L0H1M2 has no more feed-forward edges to the next level of the hierarchy which renders it totally useless. Thus, L0H1M2 can

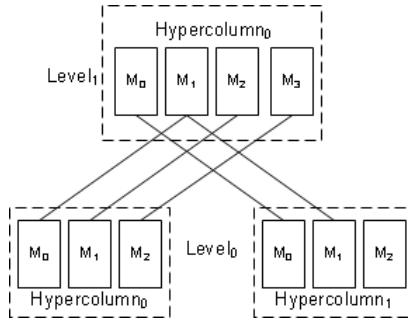


Fig. 5. Pruned graph obtained after applying the spatial correlation based algorithm on the graph shown in Figure 4

be pruned. This reduces the amount of computations required as pruning connections means less number of computations are required to recognize a shape by the minicolumns at Level 1 as the unnecessary connections are not considered for recognition.

5 Experiments and Results

To test and validate different properties of our cortical architecture and to evaluate its learning and recognition performance, we used a subset of handwritten digit images obtained from the MNIST database [17]. For this digit recognition task, we created a hierarchical network with 6 levels. We initialized this network as described in Table 2. Level 0 corresponds to the lowest level in the hierarchy. All the digits in the MNIST database are in the form of 28x28 pixel wide black and white images. Out of the 28 rows, top 2 and bottom 2 rows were always black. Thus, in our experiments, we ignored these rows to save on execution time. Each of the remaining rows becomes the input to one of the twenty four Level 0 hypercolumns.

Table 2. Detailed description of the hierarchical network created for recognition of handwritten digit images

Level	Hypercolumns (HC)	Minicolumns/HC
5	1	100
4	1	200
3	3	200
2	6	200
1	12	300
0	24	500

5.1 Experiment 1: Independent Feature Recognition

In the first experiment, we validate our feedforward information processing and learning algorithm. For this experiment, we disable the feedback processing and study how the network learns independent features from the input patterns. Since there was no

feedback, we anticipate that in Level 5 (top most level) of the hierarchy, variations of same digits will be recognized by different minicolumns. For this experiment, we took 100 handwritten digit images (10 variations of each digit) from the MNIST database and trained and tested our network with them till it achieved 100% recognition rate. In steady state, top level hypercolumn contains 89 minicolumns that learned to recognize various digit patterns present in the input dataset. 11 digit variations are pooled with some other variation of the same digit due to spatial similarities.

5.2 Experiment 2: Feedback Processing and Invariant Representation

To test how our feedback processing algorithm generates invariant representations, we use the same hierarchical network mentioned above. For the input dataset, we use the same 100 digit images (10 variations for each digit) for training as used in Experiment 1 and train the network with these images till the network achieved 100% recognition rate. At this point, we notice that there were only 10 minicolumns in the top level hypercolumn that were firing in response to the digits being exposed to the network. This means that there is just one minicolumn firing for all the different variations of the same digit. We also evaluate the resource optimization achieved through feedback processing. To do that we calculate the number of active minicolumns in the hierarchical network with and without feedback. In steady state, without feedback the network uses 3876 minicolumns while with feedback it only uses 1283 minicolumns. Thus, our feedback processing algorithm results in about 3x resource optimization.

5.3 Experiment 3: Robustness to Test Images

In this experiment, we test the robustness of our cortical network to the patterns not present in the training dataset. For this experiment we again use the same hierarchical network described above. We use 400 handwritten digits images (40 variations of each digit) training images and 40 test images (4 variations of each digit). We then train the network with the images in the training dataset till the network achieves 100% recognition rate and is in a stable state i.e. all the levels in the hierarchy have generated invariant representations for all the input digit variations. Figure 6 shows the recognition rate of the network as the number of images in the training dataset is increased from 10 to 400. For this experiment, recognition rate is defined as the percentage of the images in the test dataset that were recognized correctly.

In the future we are planning to extend our cortical architecture so that it can run on NVidia GPUs. This will let us create and test large hypercolumn based networks and will improve the recognition rates further.

5.4 Experiment 4: Inherent Fault Tolerance

This experiment validates the inherent fault-tolerant property of our cortical network. For this experiment, we use the same hierarchy as described above and use 200 handwritten digit images for training. To reduce the execution time for each epoch, we limit the feedback processing to Level 5 (top-most level) of the hierarchy only. Initially, we

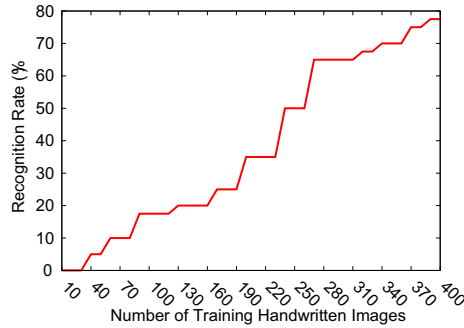


Fig. 6. Recognition rate of the network for handwritten test digit images as the number of training images is increased

train the hierarchy with all the 200 images till it achieves 100% recognition rate. At this point we corrupt 5% of the total number of minicolumns throughout the hierarchy. This was done by randomly selecting minicolumns and forcing their output to stay 0 permanently. Then we evaluate the recognition rate of the hierarchy with all the 200 training images to determine the amount of loss in recognition. Then we train the damaged hierarchy with the same training images and evaluate the peak recognition rate for the training images. We repeat this cycle multiple times corrupting 5% of the original number of minicolumns every time to observe how the hierarchy behaves as we inject more and more permanent faults. Table 3 shows the behavior of our cortical network in the presence of permanent faults.

Table 3. Evaluation of the inherent fault tolerance property of our cortical network. Initial Recognition Rate means the recognition rate (percentage) measured immediately after the faults are injected. Peak Recognition Rate means the maximum recognition rate achieved through training the damaged network.

Fault Injection Attempt	Initial Recognition Rate (%)	Peak Recognition Rate (%)
1	92	100
2	89	100
3	90	100
4	88	100
5	88	94
6	82	82
7	71	71
8	65	65

When Fault Injection Attempt is 5 that means that we have damaged 25% of the total minicolumns originally present in the hierarchy. For this attempt, after retraining the damaged hierarchy, it achieves the peak recognition rate of 94%. This is due to the fact that some of the hypercolumns ran out of the minicolumns that were idle.

As a result the features being recognized by the minicolumns that were damaged could not be relearned. This experiment also shows that as long as there are idle resources available in the network, it can recover from permanent faults.

5.5 Experiment 5: Determining Set of Sufficient Features

For this experiment, we create a 3 level hierarchical network. Each of the hypercolumns in hierarchy is initialized to have 12 minicolumns. This network is initially exposed to synthetic digit images from 0 to 9 shown in Figure 7. Each of the rows in the 7x7 digit image is exposed to one of the hypercolumns in Level 0 of the hierarchy. For example, the highlighted row in Figure 7 is exposed to the first hypercolumn in Level 0. Similarly, the second row is exposed to the second hypercolumn in Level 0 and so on. Once the hierarchical network achieves the steady state, ten of the twelve minicolumns available in the Level 2 hypercolumn fire for a unique digit image. The remaining two minicolumns keep on thrashing because they do not have any new digit to recognize.

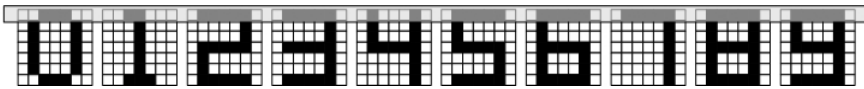


Fig. 7. Input dataset exposed to the hierarchical network to determine the set of sufficient features



Fig. 8. Set of features sufficient enough to recognize each unique shape. The features sufficient enough to identify a digit are overlaid in black on top of the actual image.

Figure 8 shows the set of features sufficient to recognize each of the ten digits. The sufficient features are shown in black. For example, to recognize a zero, the only feature that matters is the horizontal line on top of zero. Similarly, to recognize the one, the shorter horizontal line on the top is sufficient enough. In the absence of our spatial correlation based algorithm to determine the set of features sufficient for recognition of each of the digits, a total of 55 minicolumns are utilized to identify all the ten digits. With the spatial correlation based algorithm, only 33 minicolumns are required. Thus, there is an approximate 40% savings in computational resources.

Next, we again create a 3 level hierarchical network and initialize each of the hypercolumns to have 40 minicolumns. The input to this hierarchical network consists of 36 7x7 synthetic images (10 digits + 26 alphabets). To evaluate the increase in the sufficient set of features and in the number of minicolumns utilized in the presence and absence of our spatial correlation based algorithm with an increase in the unique shapes in the dataset, we exposed alphabets from A to Z to the hierarchical network along with the digits from 0 to 9.

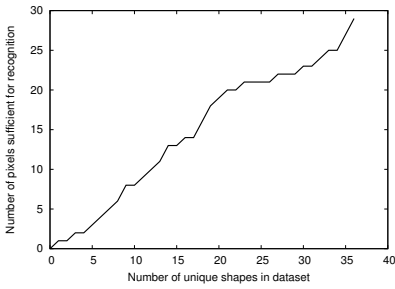


Fig. 9. Increase in the number of features sufficient to recognize all the shapes in the dataset as the number of unique shapes in the dataset is increased

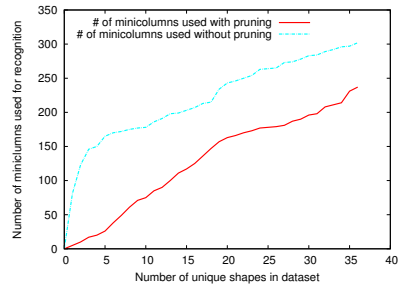


Fig. 10. Increase in the number of minicolumns required to recognize all the shapes in the dataset as the number of unique shapes in the dataset is increased

The graph in Figure 9 shows the increase in the set of features sufficient for recognition as the number of unique shapes in the dataset is increased. In this graph, the number of unique shapes in the dataset are along the x-axis while the number of sufficient features identified by the hierarchical network is along the y-axis. A linear increase in the number of sufficient features identified suggests that each of the new unique shapes adds contains at least one sufficient feature.

The graph in Figure 10 shows the increase in the number of minicolumns as the number of unique shapes in the dataset is increased. The solid line shows the number of minicolumns in all the levels of the hierarchy identified as sufficient by our spatial correlation based algorithm while the dotted line shows the total number of minicolumns used by the hierarchical network to recognize the same set of shapes if the spatial correlation based algorithm is not used. We see that the number of minicolumns required when the spatial correlation based algorithm is used is far less than the number of minicolumns used in the absence of the spatial correlation algorithm. A linear increase in the solid line in Figure 10 suggests that almost all the unique shapes in the dataset introduce a feature that is not being used by any other shape.

6 Conclusions

We describe a biological plausible learning model that implements the working of cortical columns as its basic structural and functional abstraction. We demonstrate that building models based on the properties of cortical columns can be computationally efficient as well as biologically plausible. Using these models, we can study various neocortical properties like independent feature identification, feedback, plasticity, invariant representation, and resource management. Our results show that such models are inherently tolerant to permanent faults (either in hardware or in software). Using our spatial correlation based pruning algorithm, we significantly improve the resource utilization of our hierarchical hypercolumn networks.

References

1. Nicholls, J., Martin, A., Wallace, B., Fuchs, F.: *From Neuron To Brain*. Sinauer Associates Ins., 23 Plumtree Road, Sunderland, MA, USA (2001)
2. Hawkins, J., Blakeslee, S.: *On Intelligence*. Henry Holt & Company, Inc. (2005)
3. Hirsch, J., Martinez, L.: Laminar processing in the visual cortical column. *Current Opinion in Neurobiology* 16, 377–384 (2006)
4. Aimone, J., Wiles, J., Gage, F.: Computational influence of adult neurogenesis on memory encoding. *Neuron* 61, 187–2002 (2009)
5. Markram, H.: The blue brain project. In: *SC 2006: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, vol. 53. ACM, New York (2006)
6. DARPA: Systems of neuromorphic adaptive plastic scalable electronics (synapse) (2008), <http://www.darpa.mil/dso/thrusts/bio/biologically/synapse/>
7. Clopath, C., Longtin, A., Gerstner, W.: An online hebbian learning rule that performs independent component analysis. In: *Proceedings of Neural Information Processing Systems* (2007)
8. Martinetz, T.: Competitive hebbian learning rule forms perfectly topology preserving maps. In: *International Conference on Artificial Neural Networks, ICANN*, pp. 427–434 (1993)
9. Arthur, J., Boahen, K.: Learning in silicon: Timing is everything. In: *Proceedings of Advances in Neural Information Processing Systems*. *Advances in Neural Information Processing Systems*, vol. 18, pp. 75–82 (2006)
10. Carpenter, G., Grossberg, S., Rosen, D.: Art2-a: An adaptive resonance algorithm for rapid category learning and recognition. *Neural Networks* 4, 493–504 (1991)
11. Hawkins, J., George, D.: Hierarchical temporal memory (2006), http://www.numenta.com/numenta_htm_concepts.pdf
12. George, D., Hawkins, J.: A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In: *Proceedings of International Joint Conference on Neural Networks*. *IEEE International Joint Conference on Neural Network*, vol. 3, pp. 1812–1817 (2005)
13. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. *Neural Comput.* 18, 1527–1554 (2006)
14. Mountcastle, V.: An organizing principle for cerebral function: The unit model and the distributed system. In: Edelman, G., Mountcastle, V. (eds.) *The Mindful Brain*. MIT Press, Cambridge (1978)
15. Maw, N., Pomplun, M.: Studying human face recognition with the gaze-contingent window technique. In: *Proceedings of the Twenty-Sixth Annual Meeting of Cognitive Science Society*, pp. 927–932 (2004)
16. Sigala, N., Logothetis, N.: Visual categorization shapes feature selectivity in the primate temporal cortex. *Nature* 415, 318–320 (2002)
17. Lecun, Y., Cortes, C.: The mnist database of handwritten digits (1998), <http://yann.lecun.com/exdb/mnist/>
18. Swanson, L.: Mapping the human brain: past, present, and future. *Trends in Neurosciences* 18, 471–474 (1995)
19. Mountcastle, V.: The columnar organization of the neocortex. *Brain* 120, 701–722 (1997)
20. Mountcastle, V.: Modality and topographic properties of single neurons of cat's somatic sensory cortex. *Journal of Neurophysiology* 20, 408–434 (1957)
21. Hubel, D., Wiesel, T.: Receptive fields, binocular interactions and functional architecture in cat's visual cortex. *Journal of Physiology* 160, 106–154 (1962)

22. Hubel, D., Wiesel, T.: Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology* 195, 215–243 (1968)
23. Peissig, J., Tarr, M.: Visual object recognition: do we know more now than we did 20 years ago? *Annu. Rev. Psychol.* 58, 75–96 (2007)
24. Grill-Spector, K., Kushnir, T., Hendler, T., Edelman, S., Itzhak, Y., Malach, R.: A sequence of object-processing stages revealed by fmri in the human occipital lobe. *Hum. Brain Map.* 6, 316–328 (1998)
25. Freeman, W.: Random activity at the microscopic neural level in cortex (“noise”) sustains and is regulated by low-dimensional dynamics of macroscopic activity (“chaos”). *International Journal of Neural Systems* 7, 473–480 (1996)
26. Rokni, U., Richardson, A., Bizzi, E., Seung, H.: Motor learning with unstable neural representations. *Neuron* 64, 653–666 (2007)
27. Seung, H.: Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron* 40, 1063–1073 (2003)

Computational Study of Rhythm Propagation Induced by TMS Stimuli in Different Brain Regions

Filippo Cona¹, Melissa Zavaglia¹, Marcello Massimini²,
Mario Rosanova², and Mauro Ursino¹

¹ Department of Electronics, Computer Science and Systems, University of Bologna
Via Venezia 52, 47521 Cesena, Italy

{filippo.cona2,melissa.zavaglia@unibo.it,
mauro.ursino}@unibo.it

² Department of Clinical Science, “Luigi Sacco”, University of Milan
Via Grossi 74, 20157 Milan, Italy

{marcello.massimini,mario.rosanova}@unimi.it

Abstract. Recent data [1] suggest that different regions in the brain may exhibit distinct rhythms when perturbed by Transcranial Magnetic Stimulation (TMS). Knowledge of these rhythms may be of value to understand how the brain realizes its functions and to assess brain connectivity. In this work we implemented a neural mass model [2] of three interconnected cortical regions (Brodmann Area (BA) 19 (occipital lobe), BA 7 (parietal lobe) and BA 6 (frontal lobe)) to fit the impulse responses in three ROIs during an experiment of TMS stimulation on a single subject. First, the natural rhythm of each region was mimicked acting on the local parameters, which reproduce the number of synaptic contacts among neural populations. Then, rhythm propagation from one region to another was simulated (at three different intensities of TMS stimulation) acting on infra-region connectivity parameters. Results show that the model can reproduce the natural rhythms of the three regions, and most rhythm changes induced by stimulation of another region, by using quite a simple connectivity pattern. This pattern is scarcely affected by the stimulus intensity.

Keywords: Transcranial magnetic stimulation, Neural mass models, Cortical rhythms.

1 Introduction

Brain activity normally exhibits several superimposed rhythms, which cover many frequency bands and may vary depending on the particular task or subjective status. This phenomenon has been known for many decades, starting from the early electroencephalographic recordings in the mid twenties [3]; only recently, however, brain rhythms have received sufficient attention in cognitive neuroscience. A consolidated point of view, supported by many neurophysiological data, is that these rhythms are not simply epiphenomena emerging from the complex non-linear brain dynamics, but rather they play an important role in many perceptual, motor or cognitive tasks. Researchers

found that when activity of two neuron populations is synchronized, their communication and reciprocal influence is stronger. Rhythmic activity changes have been found to play a role in memory consolidation and retrieval, in grouping and segment different features of objects, in allocating attention resources, and in processing perceptual and motor information [4; 5; 6; 7].

The previous short analysis suggests that rhythms are essential to allow efficient communication between brain regions involved in the same task. Hence, estimation of the intrinsic rhythm of regions, and of how these rhythms can be transmitted from one region to another as a consequence of inter-region connectivity, may be of the greatest importance to analyze integrative brain functioning during complex motor or cognitive functions.

A new method to achieve this information is to use the Transcranial Magnetic Stimulation (TMS) combined with EEG recordings. TMS, indeed, can be used to elicit changes in the synchronization of the brain oscillatory activities, and thus in the rhythms power [8; 9]. This technique allows the measurement of cortical reactivity and effective connectivity [10]. Moreover, TMS has been used to perturb cortical regions in order to map the different cognitive and motor functions over the brain [11] and to link these functions to characteristic oscillatory activities [12].

In a recent work, Rosanova et al. [13] observed the oscillation rate in three regions of interest (ROIs) (occipital, parietal, and frontal cortices) following TMS stimulation at different intensities in a group of healthy volunteers. Results show that the natural frequency can be directly measured in virtually any area of the cerebral cortex. Moreover, these natural rhythms seem to be transmitted, at least in part, from one region to another.

The results by Rosanova et al. [13], and more generally brain activity evoked by TMS impulses, are particularly suitable for an analysis with neurocomputational models. Indeed, computational models aim at clarifying the dynamical aspects of brain activity, to elucidate possible mechanisms of rhythm generation and rhythm transmission, and to suggest how brain regions (or specific neuronal populations within a region) can be involved in a given task. On the other hand, the response to an impulse stimulus represents the classical challenge to identify the structure of a model in a straightforward way, and to assign values to its parameters. A single impulse response, however, allows model identification in case of linear systems only. In case of strongly non-linear systems (as is certainly the case of neural dynamics) responses to impulse of different amplitudes should be tested to identify non-linearities and assess possible non-linear behaviors.

In recent years, we developed a neural mass model to study how rhythms can be generated within a cortical region and how rhythms can be transmitted from one region to another thanks to brain connectivity. The model was built starting from equations proposed by Jansen and Rit [14] and Wendling et al. [15], with inclusion of a new loop to simulate the role of fast GABA-ergic interneurons in the genesis of gamma oscillations [16]. With this new loop, the model is able to generate a gamma rhythm which can co-exist with a slower rhythm (alpha or beta) generated by a feedback loop between pyramidal neuron and slower inhibitory interneurons. Furthermore, the model is able to generate rhythms in different frequency bands, and simulate the transmission of rhythms from one region to another, by simply

modulating a few parameters which represent short-range connections within a region and inter-area long-range connectivity [16].

Aim of the present work is to test whether the model can be used to reproduce and analyze some aspects of the experimental results by Rosanova et al. [13]. In fact, model predictions (the presence of intrinsic rhythms in individual ROIs and the possibility to transmit rhythms via a few effective connections among ROIs) agree at least qualitatively with these experimental data.

Hence, the present study was designed with the following main purposes:

- i) To analyze whether the response of individual ROIs to direct TMS stimulation can be simulated with sufficient accuracy with the model, by modifying just a few internal parameters of that region. This means that we are looking for different cortical modules, which share the same basal structure but have different parameter values, and are able to reproduce different natural rhythms experimentally observed;
- ii) To analyze whether a model of interconnected ROIs can at least approximately explain how natural rhythms can be transmitted or modified as a consequence of inter-region connections;
- iii) To analyze whether the response to TMS and the estimated parameters vary significantly as a function of stimulus intensity, in order to emphasize the possible role of non-linearities.

In this work, we simulated the behavior of Brodmann Area (BA) 19 (occipital lobe), BA 7 (parietal lobe) and BA 6 (frontal lobe) with a network of three interconnected regions. Parameters are given to reproduce the effect of TMS stimulation at three different intensities in one representative subject.

2 Methods

In this section we will describe the experimental setup (2.1), the cortical source reconstruction procedure (2.2), the neural mass model used for a single area (2.3), the extension for multiple interconnected areas (2.4), the simulation of TMS/EEG experiment using the model and the fitting procedure (2.5).

2.1 Experimental Data Recording (TMS/EEG)

During the experiment, subjects were lying on an ergonomic chair, relaxed, and with their eyes open and looking at a fixation point on a screen. A focal bipulse, figure-of-eight coil with 60mm wing diameter driven by a biphasic stimulator (eXimia TMS Stimulator; Nexstim) was used to stimulate the subjects' cortex. Three cortical sites (middle or superior occipital gyrus, superior parietal gyrus, and middle or caudal portion of the superior frontal gyrus) were selected based on an atlas of brain regional anatomy [17], anatomically identified on a T1-weighted individual MRI (resolution 1 mm) acquired with a 1 T Philips scanner and were targeted by means of a Navigated Brain Stimulation (NBS) system (Nexstim). We recorded high-density EEG using a TMS-compatible 60-channel amplifier (Nexstim) which gates the TMS artifact and prevents saturation by means of a proprietary sample-and-hold circuit [18]. The EEG

signals, referenced to an additional electrode on the forehead, were filtered (0.1–500 Hz) and sampled at 1450 Hz with 16-bit resolution. Two extra sensors were used to record the electrooculogram. In most cases, no TMS-induced magnetic artefacts were detected, and in all cases, the EEG signals were artefact-free starting from 8 ms after the stimulus. TMS trials containing noise, muscle activity, or eye movements were automatically detected and rejected. The event related potentials were obtained by averaging across all the trials of each session (100–200 per session). More technical details on the procedure can be found in Rosanova et al. [13]. In each subject, we stimulated every cortical area at eight different TMS intensities (range, 20–160 V/m). Firstly we defined the EEG-threshold for each subject (on average around 50 V/m), then we used the three intensities above this threshold (medium, medium/strong and strong) for the analysis with the model. The TMS-evoked Potentials (TEPs) under the threshold were not significant with respect to the baseline activity and so they could not be used to attempt a reliable parameter fitting (see Section 3 Results).

In this work, we focus on a single subject in order to analyze the effect of the different stimulation intensities. The fitting of the whole data set will be the subject of a future work.

2.2 Cortical Sources Reconstruction

Source modelling was performed following a multiple step procedure: the free licence package SPM [19] was used to create the cortical mesh by adapting an average Montreal Neurological Institute (MNI) cortex to the subject's MRI data; skull and scalp meshes were also co-registered with EEG sensors positions into the subject's MRI space; a 3-spheres BERG method was obtained to calculate the Lead Field Matrix by using the free access Brainstorm software package [20]; the inverse solution was calculated on a single trial basis by applying an empirical Bayesian approach with estimation of covariance components using Restricted Maximum Likelihood [21]. In order to compute the overall current evoked by TMS in different cortical areas, cortical sources were attributed to different Brodmann areas using an automatic tool of anatomical classification [22]. Currents recorded within each area were cumulated in order to produce a new time series.

2.3 Model of a Single Cortical Area

The model of a cortical region consists of four neural populations, which represent pyramidal neurons, excitatory interneurons, and inhibitory interneurons with slow and fast synaptic kinetics ($GABA_{A,slow}$ and $GABA_{A,fast}$ respectively). Each population represents a group of neurons of the same type, which approximately share the same membrane potential and so can be lumped together. All populations are described with a similar mathematical formalism. Briefly, each population receives an average postsynaptic membrane potential (say v) from other neural populations, and converts this membrane potential into an average density of spikes fired by the neurons. In order to account for the presence of inhibition (when potential is below a given threshold) and saturation (when potential is high) this conversion is simulated with a static sigmoidal relationship. Moreover, each population sends synapses to other populations (or, in case of pyramidal neurons, to other regions too). Each synaptic kinetics is described with a second order system, but with different parameter values.

In the following, a quantity which belongs to a neural population will be denoted with the subscript p (pyramidal), e (excitatory interneuron), s (slow inhibitory interneuron) and f (fast inhibitory interneuron). To model a whole cortical region, the four populations are connected via excitatory and inhibitory synapses, with impulse response $h_e(t)$, $h_s(t)$ or $h_f(t)$, assuming that pyramidal neurons and excitatory interneurons synapses have similar dynamics. The average numbers of synaptic contacts among neural populations are represented by eight parameters, C_{ij} (see Fig. 1), where the first subscript represents the target (post-synaptic) population and the second subscript refers to the pre-synaptic population. These connections agree with those proposed by Wendling et al. [15] but with the addition of the new self-loop C_{ff} . The model is displayed in Fig. 1. For more details see Ursino et al. [16].

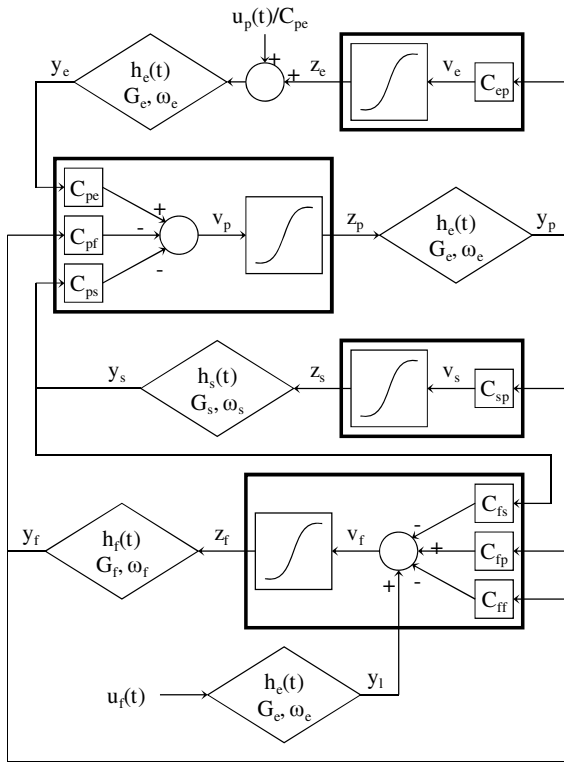


Fig. 1. Model layout of a single cortical region

2.4 Model of Connectivity among Areas

To simulate cortical connectivity between two regions (the pre-synaptic and post-synaptic regions will be denoted with the superscript k and h , respectively), we assumed that the average spike density of pyramidal neurons of the pre-synaptic area affects the target region via a weight factor, W_j^{hk} (where $j = p$ or f , depending on whether the synapse targets pyramidal neurons or GABA_{A,fast} interneurons) and a time delay D^{hk} . This is

achieved by modifying the membrane potential of the target region, with the time dynamics of an excitatory synapse. Long range synapses which target to slow inhibitory interneurons or to excitatory interneurons have not been considered since they have a minor role in model dynamics [16].

2.5 Simulation of TMS/EEG Experiment and Parameters Fitting

In order to simulate the TMS/EEG experiment described above, we implemented a model of connectivity among three cortical regions. These regions wish to simulate the Brodmann Area (BA) 19 (occipital lobe), BA 7 (parietal lobe) and BA 6 (frontal lobe).

An automatic fitting between simulated EEG and real data has been achieved in the time domain. In particular, we focused attention on the 200 ms following the TMS impulse. Experimental time series were compared with membrane potentials of pyramidal neurons simulated with the model. Since the two quantities have a different scale, all experimental time series were multiplied by a constant gain to have the same scale as the simulated signals. It is worth noting that, in the present model, we used the sum of postsynaptic potentials of pyramidal neurons to calculate the source of the EEG signal. Although this assumption is usually adopted in neural mass models [14; 15] recent studies suggest that EEG waves are generated by synaptic currents [23; 24]. We think that the use of potentials instead of currents may be acceptable in the present study, since we are especially interested in the frequency content of activity evoked by the TMS pulse. Moreover, the model works in the linear region of the sigmoidal function, where current and potential are almost proportional.

In the present work, the fitting procedure has been performed only in a frequency range above 8 Hz (i.e., theta and delta rhythms have been excluded, since they probably require a more complex model including thalamic regions, see Section 4 Discussion). As a consequence, the real TEPs were preprocessed with a high-pass filter (Chebyshev 2nd type with cutoff frequency $f_t = 8\text{Hz}$). Furthermore, oscillations in the experimental signals occurring in the first 8 ms after the perturbation were also neglected, because they can be affected by artefacts [13]. After fitting, model and real signals were also compared in the time frequency domain. To this end, time frequency maps were obtained using the continuous wavelet transform with Morlet wavelets [25].

The fitting procedure has been subdivided into two steps:

Step 1. In the first step, we fitted the impulse response of a single region when the same region receives the TMS stimulus (this step was repeated three times, once for each BA). The effect of the TMS stimulus in the single cortical area was simulated as a step change (say Δy_p) in the membrane potential of pyramidal cells, in accordance with other TMS implementations in neural models [26]. The estimated parameters were the synaptic contacts among the neural populations (C_{ij}) and the intensity of the stimulus Δy_p . To reduce the number of variables for the fitting, we used as free variables only those internal connection strengths (C_{ps} , C_{fp} , C_{pf} , C_{ff} , C_{fs} , see Table 2) that most influenced the frequency content of the model output, according to our previous study [16], for a total of 6 parameters per ROI. The optimization procedure was a combination of a Genetic Algorithm (GA) (for a similar application of GA to neural mass models see Cona et al. [27]) and the simplex method (Matlab's `fminsearch`). We used a GA in order to arrive at an optimal solution, independently of the initial guess [28]. To facilitate the convergence of the GA we used Dynamic Time Warping (DTW) [29] to compare the simulated TEPs

with the experimental ones. DTW has been used as a generalization of the Mean Square Error (MSE) because it is less influenced by time shifts and by limited deformations when comparing waveforms. The simplex method was applied once every 50 generations of the GA, in order to explore various local minima, and as the final step of the fitting procedure.

Since activities evoked by TMS at all stimulation intensities above threshold were quite repeatable in the frequency domain (i.e., TMS evoked similar natural rhythms), this step was applied only once to the data obtained with the medium stimulation and the results were thereafter used in the second step for all the three stimulation intensities. In other words, across the three stimulation intensities, we used the same internal architecture for the three regions and varied only the inter-region connectivity and the amplitude of the input.

Table 1. Model parameters that do not change during the optimization

Parameter	Value
ω_e (rad/s)	75
ω_s (rad/s)	30
ω_f (rad/s)	75
G_e (mV)	5.17
G_s (mV)	4.45
G_f (mV)	57.1
C_{ep}	5
C_{pe}	25
C_{sp}	60
e_0 (Hz)	2.5
r (mV^{-1})	0.56

Step 2. In the second step we used the same fitting algorithm (GA and simplex method) to find a unique cortical connectivity pattern that could describe the experimental TEPs of all regions both directly and indirectly triggered by TMS (i.e., all 9 signals simultaneously). This time, the algorithm acted not only on the number of synaptic contacts within each region, but also on the inter-regional connectivity strengths (directed to pyramidal cells, W_p , and to GABA_{A,fast} interneurons, W_f , for a total of 12 free parameters) and on their time delays, D^{hk} . The delays between any pair of regions were forced to be equal, thus reducing their number from 6 to 3, resulting in 15 more free parameters (33 parameters in total). The fixed parameters are found in Table 1 (note that these parameters are equal for all of the ROIs).

The results from the first step were used here as the starting points for the internal parameters in order to boost the convergence. The internal parameters and the conduction delays were forced to be equal for all the stimulation intensities in order to preserve the regions architecture (see Table 2). In this second step we used MSE as the error function instead of DTW, which is more computationally expensive, since the initial guess was sufficiently accurate. To handle the multi-objective optimization with the GA (we had to find a unique set of parameters for 9 different signals) we followed a particular strategy that makes use of more than a cost function. More precisely, we ranked every set of parameters with 9 ‘specific’ cost functions, that rewarded those individuals that best fit especially one out of the 9 signals, and 1 ‘mixing’ function that rewarded those individuals that best fit the 9 signals altogether. The probabilities for each individual to enter the

mating pool (individuals who will produce children) are equally distributed between the 10 cost functions, so an individual that fits one of the 9 signals very well has nearly the same probability to reproduce itself as an individual that fits all signals quite well. In this way the 9 signals are fitted in parallel with the goal of reaching a global minimization. The output of the GA is the set of parameters given by the individual with the lowest value for the ‘mixing’ cost function. For the simplex method we used a single cost function which is the ‘mixing’ cost function of the GA.

Table 2. Model parameters that do not change across the three stimulation intensities

Parameter	BA 19	BA 7	BA 6
C_{ps}	54	57	31
C_{fp}	81	98	137
C_{fs}	0.1	39	21
C_{pf}	4.7	10.5	11.5
C_{ff}	16	16	17.8
$D^{19,X}$ (ms)	-	1	8.3
$D^{7,X}$ (ms)	1	-	16.6
$D^{6,X}$ (ms)	8.3	16.6	-

3 Results

In this section we will show the general behavior of the three BAs by presenting the data from the medium stimulation intensity in detail (3.1) and how data change when the intensity grows or decreases (3.2).

3.1 Parameter Fitting of the Data from the Medium Stimulation Intensity

Real and simulated signals were compared both in time and frequency domains. In particular, Figs. 2-4 display the time patterns and the time-frequency maps of the simulated and real signals in response to TMS stimulation on BA 19 (Fig. 2), on BA 7 (Fig. 3) and on BA 6 (Fig. 4). Results show that the model can reproduce the main experimental patterns of cortical activity quite satisfactorily.

The main result is that each region exhibits a different intrinsic rhythm, and this rhythm exhibits evident changes as a consequence of the stimulation of another region. The model can explain both these aspects, ascribing the first to the internal parameters of the region, and the second to the mutual long-range connections among regions. Focusing on BA 19, one can observe that this region exhibits an activity mainly in the alpha range when it is directly stimulated by TMS, although with components also in the beta and gamma ranges (Fig. 2), while it oscillates in the beta and in the gamma range respectively when BA 7 (Fig. 3) and BA 6 (Fig. 4) are stimulated. BA 7 exhibits an activity in beta range when directly stimulated (Fig. 3), while it oscillates mostly in alpha and gamma range respectively when the BA 19 (Fig. 2) and BA 6 (Fig. 4) are stimulated. BA 6 oscillates mostly in gamma range when it is stimulated by the TMS (Fig. 4), and it oscillates in beta range and in alpha range respectively when BA 7 (Fig. 3) and BA 19 (Fig. 2) are stimulated.

The parameters peculiar to this simulation (TMS intensity and inter-regional connections) are shown in Table 3.

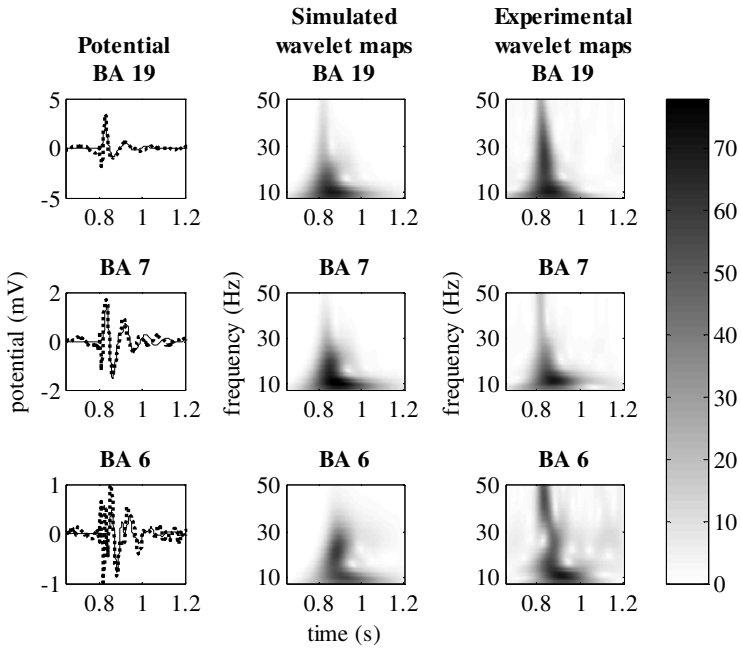


Fig. 2. Stimulation of BA 19. Simulated (solid line) and experimental time responses (dashed line) are shown in the first column. The second and third column show the simulated and the experimental time-frequency maps, respectively.

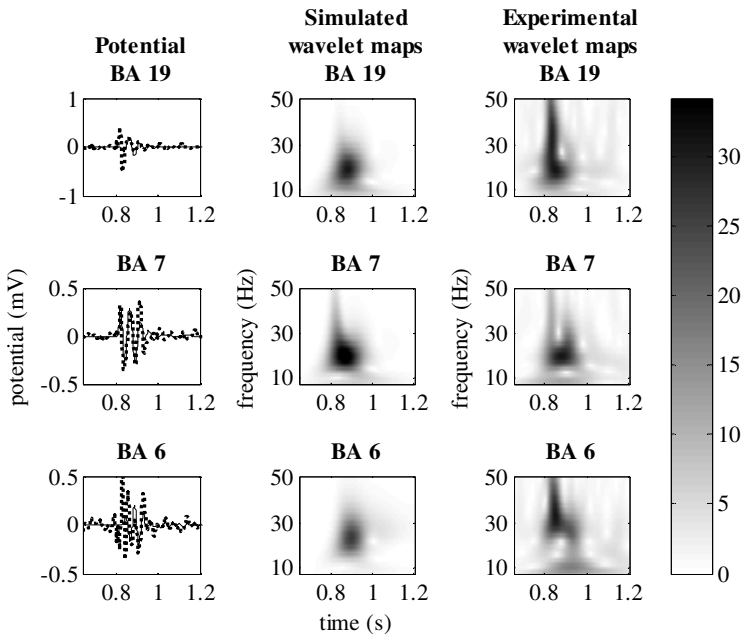


Fig. 3. Stimulation of BA 7. The panels represent the same quantities as in Figure 2.

Table 3. Model parameters peculiar to the fitting of the data from the medium stimulation intensity

Parameter	BA 19	BA 7	BA 6
Δy_p	54	57	31
$W_p^{19,X}$ (ms)	-	0	16.5
$W_p^{7,X}$ (ms)	94.5	-	0
$W_p^{6,X}$ (ms)	0.57	25.5	-
$W_f^{19,X}$ (ms)	-	81	24.5
$W_f^{7,X}$ (ms)	75.5	-	11.5
$W_f^{6,X}$ (ms)	0	0	-

3.2 Generalization to All Stimulation Intensities

The previous results suggest that each region exhibits a different natural frequency when stimulated with TMS. This intrinsic rhythm can be ascribed to the internal parameters of the region. Moreover, these rhythms can be propagated from one region to another thanks to the extrinsic connectivity.

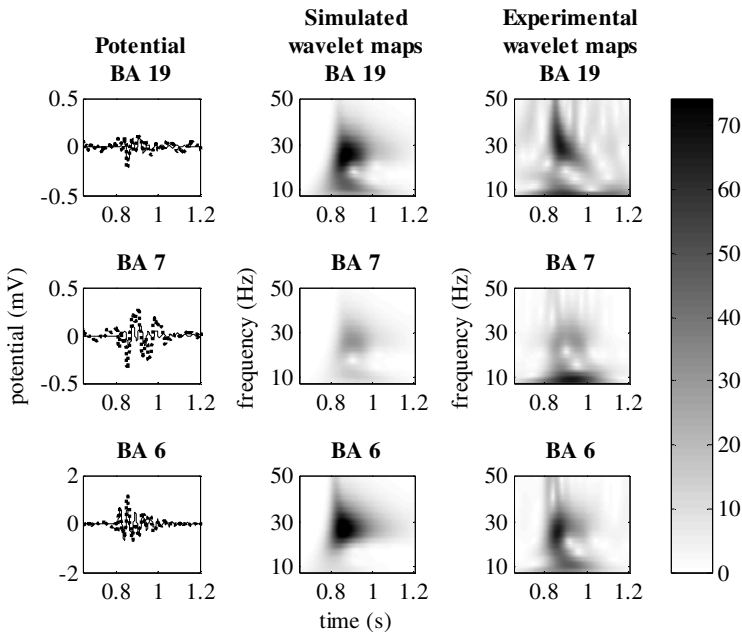


Fig. 4. Stimulation of BA 6. The panels represent the same quantities as in Figure 2.

The next step was to assess the effect of the stimulation intensity of TMS response. To this end, we fitted the data obtained using two stronger intensities (medium/strong and strong) as described in section “Methods”.

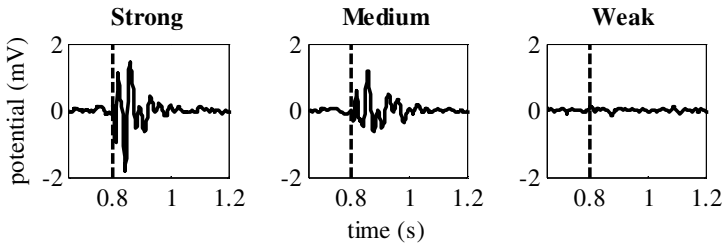


Fig. 5. Response of BA 6 when it is stimulated with strong (160 V/m), medium (120 V/m) and weak (60 V/m) stimulation intensities. The dashed lines indicate the instant in which the TMS impulse is administered.

In the present work we did not fit the data obtained with weaker stimulations. The reason is that TMS stimulation at low intensity did not evoke significant EEG changes compared with baseline activity. Fig. 5 shows the TEP in BA 6 when this region is stimulated with different TMS intensities. One can see that the TEP decreases with the intensity of the TMS stimulus, but in a non-linear fashion: the response becomes indistinguishable from baseline activity below a certain threshold.

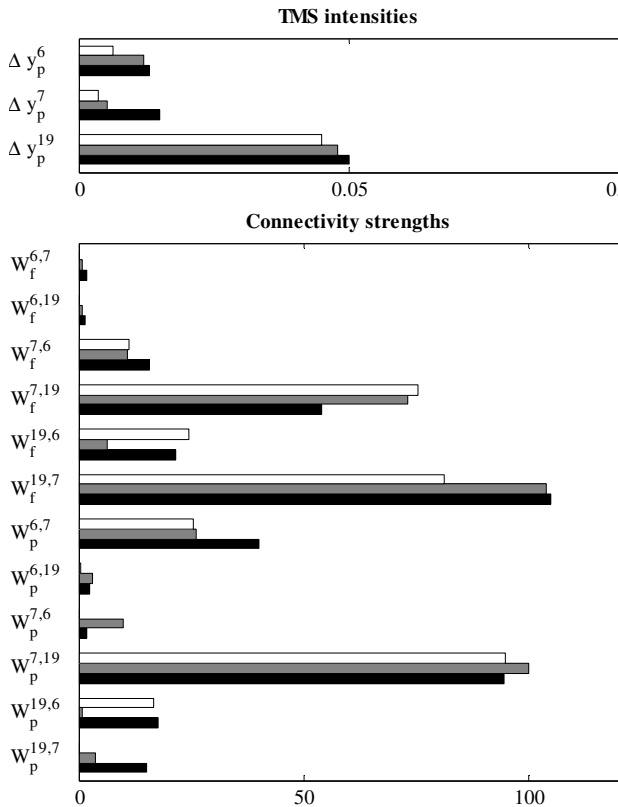


Fig. 6. Comparison of model parameters obtained from medium (white bars), medium/strong (gray bars) and strong (black bars) stimulation intensity. The upper panel shows the inputs Δy_p to BA 6, BA 7 and BA 19. The lower panel shows the inter-regional connection strengths.

The results are summarized in Fig. 6. The upper panel shows that the estimate impulse actually increases with the strength of the TMS stimulation. The lower panel shows that the inter-region connectivity remains quite stable across the different intensities, i.e. it is not easy to detect a clear dependence of connectivity on the impulse strength.

4 Discussion

The main objective of the present work was to investigate whether a recent neural mass model of interconnected regions [16], can explain the patterns of neural rhythms evoked by TMS, in three different cortical regions (occipital, parietal and frontal). Results are reliable and underline the following fundamental aspects:

- i) The impulsive response evoked in a cortical region via a TMS stimulus can be fitted quite well acting just on a few parameters internal to the region, which represent the number of synaptic connections between the neural populations involved.
- ii) Different regions exhibit different natural rhythms when directly stimulated by TMS (roughly in the alpha band for the BA19, in the beta band for the BA7 and in the gamma band for the BA6). This result, which was well evident in the former work by Rosanova et al. [13], is now explained in terms of differences in the internal connections between the neural populations, without the need to hypothesize changes in synaptic dynamics (i.e., all synapses which refer to a given class of neurons have the same dynamics in the model, independently of the cortical region). This explanation is physiologically reliable, since internal connections within cortical columns are probably different among different Brodmann areas. We suggest that these differences are reflected in differences of the natural rhythms, and these can be evoked by short impulsive perturbations as those induced by TMS. The relationship between internal connectivity among neural populations and natural rhythm represents an interesting testable prediction.
- iii) The natural rhythms in a ROI are modified if another region is stimulated. The present model can simulate these rhythm changes fairly well, ascribing them to effective connectivity among ROIs. Substantially, the main new result of this study is that a pattern of connectivity among the ROIs (targeting pyramidal neurons and/or fast interneurons) can explain how a natural frequency can be modified and/or a new rhythm can be received following stimulation of another ROI. In general, the simple connectivity pattern shown in Table 3 can mimic many of the rhythm changes observed during the experiment in the three regions (occipital, parietal and frontal). In perspective, this result may have important implications for neuroscience. On one hand, the way a natural rhythm is transmitted from one region to another may play an important role in many perceptive, motor or cognitive brain functions [30; 31; 32]. Furthermore, the observation of rhythm changes in different brain regions may provide important clues to assess brain connectivity from high resolution scalp EEG, a problem of large relevance in contemporary cognitive neuroscience.

- iv) The model can simulate the responses to different intensities (medium, medium-strong, strong) using a single set of parameters for the internal regions, and assuming only moderate changes in infra-region connectivity. This signifies that non-linear behaviors are not fully evident in this range of stimulus intensities. In particular, we could not find a clear trend of variation in connectivity strength when passing from one stimulus intensity to another, apart from a progressive increase in the input (which was an estimated parameter of the fitting algorithm). Conversely, an abrupt non-linearity is evident when the stimulus intensity falls below a threshold (see Fig. 5). We did not reproduce this non-linearity since we set the working point of the model in the central region of the sigmoid activation function (a choice shared by many other authors which used neural mass models [33]). Perhaps a best simulation on non linear activation of ROIs may be achieved in future works assuming a working point in basal condition closer to the bottom portion of the sigmoid. In this way, a significant activation arises only when TMS stimulus exceeds a given threshold, thus paralleling experimental results at low intensity. In this work we have chosen to work at the central point of the sigmoid assuming that ROIs are naturally active to some extent as a consequence of normal brain functioning. As discussed below, a threshold mechanism may also be mimicked including a thalamic module in the model.
- v) Parameter estimation seems quite robust, as demonstrated by the moderate changes in connectivity strength from one trial to the next. In particular, our results suggest, independently of the stimulus intensity, the existence of a strong feedback loop between ROI 19 and ROI 7, a significant but less strong loop between ROI7 and ROI6, and a reentrant connection from ROI6 to ROI19 (although the latter exhibits a greater variability among trials). Finally, it is observable that connections directed from pyramidal cells toward fast GABAergic interneurons are stronger on the average than those directed toward pyramidal neurons. We do not think that this result signifies that excitatory pyramidal-pyramidal connections are unimportant in brain connectivity. Rather, this result underlines that fast inhibitory interneurons play an essential role in rhythm transmission, especially at high EEG frequencies (high beta and gamma). This is probably a consequence of their fast dynamics. A similar conclusion was achieved, via a sensitivity analysis, in another modeling paper [16].

Although the results attained in the present work are quite satisfactory, the study also exhibits several limitations, which may become the target of future improvements or extensions. First, there is no warranty that the parameter values obtained in this study are unique. Probably, different combinations of parameters do exist which provide similar results. The problem of the uniqueness of parameter estimates is very complex in all non-linear fitting procedures. The solution will require the inclusion of additional knowledge, to constrain parameter estimates (for instance, the use of additional anatomical or neurophysiological knowledge, and the use of Bayesian estimation techniques).

The experiment was performed on 7 subjects, but in this pilot study we used data recorded just from one of them. Fitting to all available data will be attempted in future works. Comparison between the connectivity patterns obtained on different subjects will be of the greatest value to check the repeatability of the obtained results, and to understand which connectivity parameters are more subject dependent.

In the present study, we did not consider connections between cortical regions and the thalamus. Conversely, cortico-thalamic connections are known to play a pivotal

role in generating brain oscillations [32] as well as in the transmission of information among cortical regions. The choice of not including thalamic regions in the model was motivated by a parsimony reason: we wished to realize a parsimonious model of a TMS stimulation experiment, with a reduced number of regions and of connectivity parameters. Consequently, a single ROI in the model can be considered representative not only of cortical dynamics, but more generally of an entire cortico-thalamic circuit. Inclusion of an explicit description of the thalamus may represent a possible model extension. However, we expect that this enlarged model would require more data to fit individual parameters in both cortical and thalamic regions altogether.

An important role played by a thalamic module may be that of increasing the non linear behavior in the integrate model, especially for what concerns the mechanisms at the basis of oscillation genesis and oscillation transmission. In fact, the thalamus plays a critical role in controlling cortical activity and establishing cortico-cortical connections: this role is frequently assimilated to that of a “gating mechanism”. For instance, the different responses observed at low and medium stimulus intensities might be better explained assuming a positive loop between a ROI and a thalamic nucleus: the thalamic nucleus would emphasize the stimulus response via a feedback loop only when intensity overcomes a given threshold. A further important function of the thalamus is that of inducing low-frequency rhythms (especially in the delta and theta ranges) superimposed on the natural rhythms of the ROIs. Indeed, in the present work we neglected these low-frequency oscillations (see for instance the small power at low frequencies in Fig. 4) and focused attention only on the alpha, beta and gamma ranges. Low-frequency oscillations may be analyzed in future works after insertion of one or more a thalamic nuclei in the model. Furthermore, the thalamic neurons exhibit a further mechanism of activation (named “bursting”) i.e., they fire at high frequency when the membrane potential is depolarized from a relatively hyperpolarized value. Some authors assumed that these bursts serve as a “wake-up call” to the cortex to signal a change in the outside world. This aspect too may become relevant, for instance to signal a TMS stimulus of sufficient intensity.

References

1. Rosanova, M., Casali, A., Bellina, V., Resta, F., Mariotti, M., Massimini, M.: Natural Frequencies Of Human Corticothalamic Circuits. *J. Neurosci.* 29, 7679–7685 (2009)
2. Ursino, M., Cona, F., Zavaglia, M.: The Generation Of Rhythms Within A Cortical Region: Analysis Of A Neural Mass Model. *Neuroimage* 52, 1080–1094 (2010)
3. Berger, H.: Über Das Elekrenkephalogramm Des Menschen. *Arch. Psychiatr. Nervenkr.* 87, 527–570 (1929)
4. Fries, P., Nikolic, D., Singer, W.: The Gamma Cycle. *Trends Neurosci.* 30, 309–316 (2007)
5. Jensen, O., Kaiser, J., Lachaux, J.P.: Human Gamma-Frequency Oscillations Associated With Attention And Memory. *Trends Neurosci.* 30, 317–324 (2007)
6. Thut, G., Miniussi, C.: New Insights Into Rhythmic Brain Activity From Tms-Eeg Studies. *Trends Cogn. Sci.* 13, 182–189 (2009)
7. Wang, X.J.: Neurophysiological And Computational Principles Of Cortical Rhythms In Cognition. *Physiol. Rev.* 90, 1195–1268 (2010)
8. Brignani, D., Manganotti, P., Rossini, P.M., Miniussi, C.: Modulation Of Cortical Oscillatory Activity During Transcranial Magnetic Stimulation. *Hum. Brain Mapp.* 29, 603–612 (2008)

9. Fuggetta, G., Pavone, E.F., Fiaschi, A., Manganotti, P.: Acute Modulation Of Cortical Oscillatory Activities During Short Trains Of High-Frequency Repetitive Transcranial Magnetic Stimulation Of The Human Motor Cortex: A Combined Eeg And Tms Study. *Hum. Brain Mapp.* 29, 1–13 (2008)
10. Miniussi, C., Thut, G.: Combining Tms And Eeg Offers New Prospects In Cognitive Neuroscience. *Brain Topogr.* 22, 249–256 (2010)
11. Hallett, M.: Transcranial Magnetic Stimulation: A Primer. *Neuron* 55, 187–199 (2007)
12. Thut, G., Miniussi, C.: New Insights Into Rhythmic Brain Activity From Tms-Eeg Studies. *Trends Cogn. Sci.* 13, 182–189 (2009)
13. Rosanova, M., Casali, A., Bellina, V., Resta, F., Mariotti, M., Massimini, M.: Natural Frequencies Of Human Corticothalamic Circuits. *J. Neurosci.* 29, 7679–7685 (2009)
14. Jansen, B.H., Rit, V.G.: Electroencephalogram And Visual Evoked Potential Generation In A Mathematical Model Of Coupled Cortical Columns. *Biol. Cybern.* 73, 357–366 (1995)
15. Wendling, F., Bartolomei, F., Bellanger, J.J., Chauvel, P.: Epileptic Fast Activity Can Be Explained By A Model Of Impaired Gabaergic Dendritic Inhibition. *Eur. J. Neurosci.* 15, 1499–1508 (2002)
16. Ursino, M., Cona, F., Zavaglia, M.: The Generation Of Rhythms Within A Cortical Region: Analysis Of A Neural Mass Model. *Neuroimage* 52, 1080–1094 (2010)
17. Tamraz, J., Comair, Y.: Atlas Of Regional Anatomy Of The Brain Using Mri., Berlin (2000)
18. Virtanen, J., Ruuhonen, J., Naatanen, R., Ilmoniemi, R.J.: Instrumentation For The Measurement Of Electric Brain Responses To Transcranial Magnetic Stimulation. *Med. Biol. Eng Comput.* 37, 322–326 (1999)
19. Statistical Parametric Mapping, <http://www.fil.ion.ucl.ac.uk/spm>
20. Brainstorm, <http://neuroimage.usc.edu/brainstorm>
21. Friston, K., Henson, R., Phillips, C., Mattout, J.: Bayesian Estimation Of Evoked And Induced Responses. *Hum. Brain Mapp.* 27, 722–735 (2006)
22. Advanced Neuroscience Imaging Research Laboratory, <http://www.ansir.wfubmc.edu>
23. Avitan, L., Teicher, M., Abeles, M.: Eeg Generator-A Model Of Potentials In A Volume Conductor. *J. Neurophysiol.* 102, 3046–3059 (2009)
24. Nunez, P.L., Srinivasan, R.: Electric Fields Of The Brain: The Neurophysics Of Eeg, 2nd edn., New York (2006)
25. Tallon-Baudry, C., Bertrand, O., Delpuech, C., Pernier, J.: Stimulus Specificity Of Phase-Locked And Non-Phase-Locked 40 Hz Visual Responses In Human. *J. Neurosci.* 16, 4240–4249 (1996)
26. Esser, S.K., Hill, S.L., Tononi, G.: Modeling The Effects Of Transcranial Magnetic Stimulation On Cortical Circuits. *J. Neurophysiol.* 94, 622–639 (2005)
27. Cona, F., Zavaglia, M., Astolfi, L., Babiloni, F., Ursino, M.: Changes In Eeg Power Spectral Density And Cortical Connectivity In Healthy And Tetraplegic Patients During A Motor Imagery Task. *Comput. Intell. Neurosci.*, 279–515 (2009)
28. Holland, J.: Adaptation In Natural And Artificial Systems (1975)
29. Sakoe, H., Chiba, S.: Dynamic Programming Algorithm Optimization For Spoken Word Recognition. *Trans. On Assp.* 26, 43–49 (1978)
30. Fries, P., Nikolic, D., Singer, W.: The Gamma Cycle. *Trends Neurosci.* 30, 309–316 (2007)
31. Kaiser, J., Lutzenberger, W.: Human Gamma-Band Activity: A Window To Cognitive Processing. *Neuroreport* 16, 207–211 (2005)
32. Steriade, M.: Grouping Of Brain Rhythms In Corticothalamic Systems. *Neuroscience* 137, 1087–1106 (2006)
33. David, O., Harrison, L., Friston, K.J.: Modelling Event-Related Responses In The Brain. *Neuroimage* 25, 756–770 (2005)

Smart Growing Cells: Supervising Unsupervised Learning

Hendrik Annuth and Christian-A. Bohn

Wedel University of Applied Sciences, Wedel, Germany
{annuth,bohn}@fh-wedel.de
<http://cg.fh-wedel.de>

Abstract. In many cases it is reasonable to augment general unsupervised learning by additional algorithmic structures. Kohonens self-organizing map is a typical example for such kinds of approaches. Here a 2D mesh is superimposed on pure unsupervised learning to extract topological relationships from the training data. In this work, we propose generalizing the idea of application-focused modification of ideal, unsupervised learning by the development of the *smart growing cells* (SGC) based on Fritzsche's *growing cells structures* (GCS). We substantiate this idea by presenting an algorithm which solves the well-known problem of surface reconstruction based on 3D point clouds and which outperforms the most classical approaches concerning quality and robustness.

Keywords: Neural networks, Unsupervised learning, Self-organization, Growing cell structures, Surface reconstruction.

1 Introduction

The idea of developing the smart growing cells approach is driven by the need of an algorithm for robust surface reconstruction from 3D point sample clouds.

The demand for efficient high quality reconstruction algorithms has grown significantly in the last decade, since the usage of 3D point scans has widely been spread into new application areas. These include geometric modeling to supplement interactive creation of virtual scenes, registering landscapes for navigation devices, tracking of persons or objects in virtual reality applications, medicine, or reverse engineering.

3D points, retrieved by laser scanners or stereo cameras, introduce two vital questions. First, how can one recognize a topology of the originating 2D surfaces just from independent 3D sample points and without any other information from the sampled objects? Second, for further processing, how is it possible to project this topological information on a data structure like a triangle mesh — meeting given constraints concerning mesh quality and size?

Although this issue has intensely been tackled since the early eighties [1] a general concept that addresses all the problems of surface reconstruction has not been determined up to now. Noise contained in the sample data, anisotropic point densities, holes and discontinuities like edges, and finally, handling vast amounts of sampling data are still a big challenge.

Previous Work. The issue of surface reconstruction is a major field in computer graphics. There are numerous approaches with different algorithmic concepts. In [6] and [12] an implicit surface is created from point clouds which then is triangulated by the marching cubes approach. [2] and [14] reduce a delaunay tetrahedralization of a point cloud until the model is carved out. Approaches like [16] or [7] utilize techniques based on the Bayes' theorem.

In the area of artificial neural networks a famous work is [13]. They propose the *Self Organizing Map* (SOM) which iteratively adapts its internal structure — a 2D mesh — to the distribution of a set of samples and enables clustering or dimensionality reduction of the sample data. While a SOM has a fixed topology, the growing cells structures concept [34] allows the network for dynamically fitting its size to the sample data complexity. SOM and GCS are suitable for processing and representing vector data like point samples on surfaces. [5] uses a SOM and [18] and [20] a GCS for the purpose of surface reconstruction. Further improvements are made by [8] where constant Laplacian smoothing [17] of surfaces is introduced, and in [9] the curvature described by the input sample distribution is taken to control mesh density. In [11] the GCS reconstruction process is further enhanced in order to account for more complex topologies. [10] use several meshes of the same model for a mesh optimization process, and [19] present a concept for combining common deterministic approaches and the advantages of the GCS approach.

In the following, we outline the basis of our approach — the growing cells structures — and then derive our idea of the smart growing cells, which matches the specific requirements of reconstruction. Afterwards, an analysis is compiled discussing mesh quality and performance of our approach, and finally, we close with a summary and a list of future options of this work.

2 Reconstruction with Smart Growing Cells

Classical growing cells approaches for reconstruction tasks are based on using the internal structure of the network as a triangulation of the object described by a set of surface sample points. A 2D GCS with 3D *cells* is trained by 3D sample points. Finally, the cells lie on the object surface which the 3D points represent. The network structure — a set of 2D simplices (triangles) — is directly taken as triangulation of the underlying 3D object. The reason for using a GCS for reconstruction are its obvious advantages compared to deterministic approaches.

- They can robustly handle arbitrary sample set sizes and distributions which is important in case of billions of unstructured points.
- They are capable of reducing noise and ply discontinuities in the input data.
- They are capable of adaption — it is not required to regard all points of the sample set on the whole. Incrementally retrieved or stored samples can be used for retrain without starting the triangulation process from scratch.
- They guarantee to theoretically find the best solution possible. Thus, approximation accuracy and mesh quality are automatically maximized.

Place k reference vectors $\mathbf{c}_i \in \mathbb{R}^n$, $i \in \{0..k-1\}$ randomly in input space.

repeat

Chose sample $\mathbf{s}_j \in \mathbb{R}^n$ randomly from the input set.

Find reference vector \mathbf{c}_b closest to \mathbf{s}_j (“best matching” or “winning unit”).

Move \mathbf{c}_b into the direction of \mathbf{s}_j according to a certain strength ϵ_{bm} , like $\mathbf{c}_b^{\text{new}} = \mathbf{c}_b^{\text{old}}(1 - \epsilon_{bm}) + \mathbf{s}_j \epsilon_{bm}$.

Decrease ϵ_{bm} .

until $\epsilon_{bm} \leq$ certain threshold ϵ_0 .

Fig. 1. The general unsupervised learning rule

Nevertheless, these advantages partly clash with the application of reconstruction. On the one hand, discontinuities are often desired (for example, in case of edges or very small structures on object surfaces). On the other hand, smoothing often destroys important aspects of the model under consideration (for example, if holes are patched, if separate parts of the underlying objects melt into one object, or if the object has a very complex, detailed structure). In such cases, GCS tend to generalize which mostly lets vanish visually important features which the human is sensitized to.

The presented smart growing cells approach accounts for these application-focused issues and emphasizes that modification of the general learning task in the classical GCS is suitable for many novel application fields.

2.1 Unsupervised Learning and Growing Cells Structures

General unsupervised learning is very similar to *k-means clustering* [15] which is capable of placing k n -dimensional reference vectors in a set of n -dimensional input samples such that they may be regarded as means of those samples which lie in the n -dimensional Voronoi volume of the reference vectors. Unsupervised learning is based on iteratively adapting reference vectors by comparing them to the n -dimensional input samples set, described with the algorithm in Fig. 1.

Surface reconstruction with pure unsupervised learning would place a set of reference vectors on the object but does not determine information about the underlying surface topology, which leads to the Kohonen Self Organizing Map.

The Kohonen self organizing map is based on reference vectors which now are connected through a regular 2D mesh. The general unsupervised learning rule is extended to account for the direct neighborhood of a best matching unit with the loop from Fig. 2.

Insertion into the general unsupervised learning algorithm (after moving of \mathbf{c}_b) in Fig. 1 leads to the phenomenon that the reference vertices now are moved by accounting for the regular 2D mesh topology of the SOM. Training a plane-like sample set leads to an adaption of the SOM grid to this implicit plane — the sample topology is recognized and finally represented by the SOM mesh.

```

for all  $\mathbf{c}_{nb} \in \text{neighborhood of } \mathbf{c}_b$  do
  Move  $\mathbf{c}_{nb}$  in the direction of  $\mathbf{s}_j$  according to a certain strength  $\epsilon_{nb}$ , like
   $\mathbf{c}_{nb}^{\text{new}} = \mathbf{c}_{nb}^{\text{old}}(1 - \epsilon_{nb}) + \mathbf{s}_j\epsilon_{nb}$ .
  Decrease  $\epsilon_{nb}$ .
end for

```

Fig. 2. Accounting for the cell topology by introducing the neighborhood of a winning unit in the general training from Fig. 1.

Nevertheless, the mesh size of a SOM is fixed and cannot adjust to the sample structure complexity. The growing cells structures overcome this drawback.

The Growing cells structures — to a certain degree — may be seen as SOM which additionally are capable of growing and shrinking according to the problem under consideration which is defined by the sample distribution. This mechanism is based on a so called *resource term* contained in every reference vector and which — in the original approach — is a simple counter. It counts how often a certain reference vector has been detected being a best matching unit. A big counter value signalizes the requirement for insertion of new reference vectors.

With a GCS one could train a sample set lying on a certain object surface and the network structure would fit the object surface at a certain approximation error. The problem is that in reconstruction tasks sample distributions are often not uniform. The represented surfaces usually contain discontinuities like sharp edges and holes, and the objects to be reconstructed are not that simple like a plane or a tetrahedron. The latter usually are chosen as initial networks and can hardly adapt to complex topologies, since only objects which are homeomorphic the start object can be represented satisfactorily.

Thus, general unsupervised learning must evolve to a kind of constrained unsupervised learning which detects and adapts to certain structures which the sample set implicitly contains.

2.2 Smart Growing Cells

Let's have a "biological view" on a network of neural cells. Here, growing cells would expose a typical unicellular organism — all cells are identical, they possess the same abilities.

Now, smart growing cells break this limitation, like it happens in "real world". During training they change their capabilities according to the tasks they will have to fulfill and which is implicitly determined by the training input — in case of the SGC, the underlying sample distribution. This changes the abilities of the general unsupervised neural network significantly. In contrast to common GCS networks where the cell behavior is limited to rules that concern Euclidian distance only, the behavior of an SGC network can precisely be modelled but without breaking the favorable, iterative, unsupervised characteristics of the GCS training rules.

The SGC basic structure is identical to general GCS, i.e., there are n -dimensional cells which — from now on — are termed *neural vertices* connected by *links* through an

repeat

for $j = 1$ to k_{del} **do**

for $i = 1$ to k_{ins} **do**

Select sample s from point cloud randomly, find closest neural vertex and move it together with neighbor vertices towards s .

Increase signal counter at s (the resource term mentioned above) and decrease the signal counters of all other vertices.

end for

Find best performing neural vertex (with highest signal counter value) and add new vertex at this position (see Fig. 4).

end for

Find worst performing neural vertices, delete them and collapse regarding edges (see Fig. 4).

until certain limit like approximation error, or number of vertices is reached.

Fig. 3. Classical growing cells structures algorithm

m -dimensional topology. Let $n = 3$ since neural vertices are directly taken as vertices of the triangulation mesh and $m = 2$ since we aim at 2D surfaces to be reconstructed.

The main training loop is outlined in Fig. 3. Here k_{del} and k_{ins} are simple counter parameters defined below (see section 2.3). Movements of vertices and their neighbors slightly differ from the classical SOM. Again, there are two parameters for the learning rates, ϵ_{bm} for the winner and ϵ_{nb} for its neighbors, but these are not decreased during learning since vertex connections automatically become smaller together with the learning rates. For drawing the neighboring vertices, a smoothing process like described in [8] and [17] is applied, which replaces the classical movement, and which makes the adaption of the topology more robust.

As initial network, usually a tetrahedron or a plane with random vertices is suitable. Two operations enable the network to grow and shrink — “vertex split” and “edge collapse”.

The vertex split operation adds three edges, two faces, and a new neural vertex. The longest edge at the neural vertex with the highest resource term is split and a new vertex is added in the middle. The signal counter value is equally spread between the two vertices (see Fig. 4).

Edge collapse removes all neural vertices with resource terms below a certain threshold r_{min} together with three edges and two connected faces (see Fig. 4). The determination of the edge to be removed is driven by connectivity irregularities as proposed in [8].

It follows the adaption of the cell behavior driven by the application needs of surface reconstruction. It leads to our proposal of lifting cell capabilities above that of general unsupervised learning described in the following paragraphs.

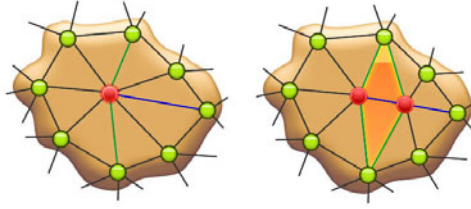


Fig. 4. Neural vertex split operation (read from left to right) to increase mesh granularity locally, and edge collapse (read from right to left) to shrink mesh locally

A) Cell Weeding. Deleting neural vertices which are not part of a sound underlying mesh structure is the most important new training rule of the SGC approach. It is essential for giving the network the chance of adapting to any topology despite of its initial topology (overcoming the homomorphic restriction). Before the edge collapse operation is applied at a vertex, it will be tested if the vertex is contained in a *degenerated mesh region* (definition follows below). If so, an *aggressive cut out* of the vertex and its surrounding vertices is started.

It has been shown that degeneration of a part of a mesh serves as perfect indicator for a mesh topology which does not fit the underlying sample structure correctly. For example, consider a region where sample densities equal zero. Although vertices are not directly drawn into it by training adjustment, their neighbors may be moved there through their mesh connections. Due to their resource terms, these vertices will be deleted by edge collapse operations, but their links remain and mistakenly represent the existence of some topology. In this case, the structure of the links is degenerated, i.e., it usually shows a surpassing number of edges with *acute-angled*¹ vertices (see Fig. 5).

The reason for terming this deletion "aggressive" are the triggering properties which are quite easy to match — suspicious neural vertices will be cut out early.

A *Criterion for degenerated mesh regions* is already proposed in [11] where a large area of a triangle is taken as sign for a degenerated mesh structure. But it has been shown that this criterion warns very late. Also, anisotropic sample densities are mistakenly interpreted as degenerated mesh regions. Our proposal is a combination of *vertex valence*², triangle quality, and quality of neighboring vertices. If all of the following conditions hold, deleting of the mesh structure at that vertex is triggered.

1. Vertex valence rises above a certain threshold $n_{degvallence}$.
2. Vertex is connected to at least $n_{degacute}$ acute-angled triangles.
3. Vertex has more than n_{degnb} neighbors for which conditions (1) or (2) hold.

¹ A triangle is termed acute-angled if the ratio of its area and the area which is spanned by a second equilateral triangle built from the longest edge of the first lies below a certain threshold ϵ_{acute} .

² Vertex valence is the number of connected vertices.

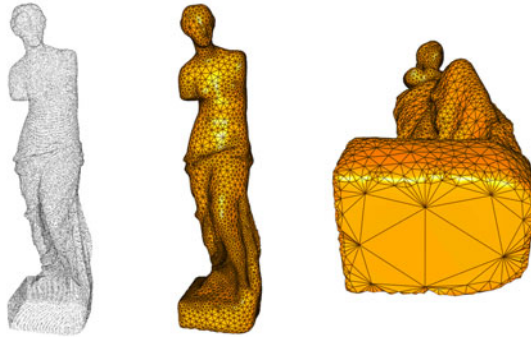
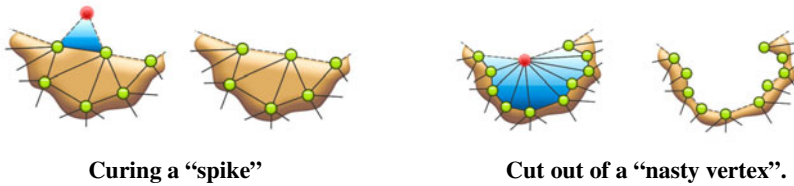


Fig. 5. Statue’s bottom is not represented by samples. On the right, the acute-angled triangles expose a degenerated mesh region.



Curing a “spike”

Cut out of a “nasty vertex”.

Fig. 6. Two types of unwanted vertices and their extinction

The latter condition says that deletion is only started if at least one or two neighbors have the same inconsistencies in their local mesh structure. This is reasonable since single degenerated vertices do not necessarily expose a problem but may arise by accident.

Curing boundaries after weeding is needed, since, after an aggressive extinction of a neural vertex and its surrounding faces has happened, usually a boundary will be left which may consist of unfavorable mesh structure elements. Curing finds these structures along the boundary and patches them discriminating between four cases, described in the following.

i) The Spike. A boundary vertex with a valence of 2 (see left part of Fig. 6) is termed a spike. This type of vertex is very unlikely to support a correct reconstruction process since it will be adjusted to an acute-angled triangle after few iteration steps. A spike must be deleted completely.

ii) The Nasty Vertex. A nasty vertex is a neural vertex with at least $n_{nastyacute}$ acute-angled triangles and/or triangles with a valence greater than $n_{nastyval}$ (see Fig. 6). It is suspected to be part of a degenerated mesh region and is deleted.

iii) The Needle Eye. A needle eye is a neural vertex that is connected to at least two boundaries (see Fig. 7 on the left). At these locations the mesh does not have a valid mesh structure. To delete a needle eye, all groups of connected faces are determined — the group with the most faces survives, all others are deleted.

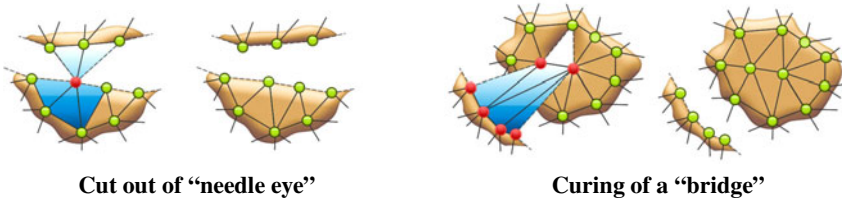


Fig. 7. Two types of unwanted connections between separate topologies

iv) *The Bridge*. A bridge is very likely to be part of a degenerated mesh region. A mesh with a hole consisting of three vertices would soon be closed by a coalescing process (see section 2.2). This is not allowed if exactly one of the edges of this hole would additionally be connected to a face (which we term a “bridge”, see Fig. 7) since an invalid edge with three faces would arise. The entire bridge structure is deleted and the hole will be closed by generating a new face.

B) Coalescing Cells. Like a mesh can be split through deletion of vertices, it must also be possible to merge two mesh boundaries during training. For that, a coalescing test is accomplished each time a vertex at a mesh boundary is moved.

The coalescing test determines if two boundaries are likely to be connected to one coherent area. For that, a sphere is created with the following parameters. Given the neighboring boundary vertices v_1 and v_2 of c_b , then let $c = 1/2(v_1 + v_2)$. A *boundary normal* n_c is calculated as the average of all vectors originating at c and ending at neighbors of c_b , where v_1 and v_2 are not taken into account. The boundary normal can be seen as a direction pointing to the opposite side of the boundary. We define a sphere with the center at $c + n_c r$ with radius r as the average length of the edges at c_b .

The coalescing condition at two boundaries hold, i.e., merging of the boundaries containing c_b and q on the opposite side happens, if

- q is contained in the defined sphere, and
- scalar product of the boundary normals at c_b and q is negative.

The Coalescing process is required since after detecting the neural vertex q to be connected with c_b , the according faces must be created starting with one edge from c_b to q . There are two cases which have to be considered.

i) *Corner*. A corner of the same boundary arises when c_b and q have one neighboring vertex in common (see Fig. 8). A triangle of the three participating vertices is created.

ii) *Long side*. Here, two boundaries appear to be separated. After determining the new edge, there are four possibilities for insertion of a new face containing the edge (see second picture in Fig. 9). The triangle with edge lengths which vary fewest is taken in our approach (see third picture in Fig. 9) since it is the triangle with the best features concerning triangle quality. Finally, to avoid a needle eye, a further triangle must be added — again, the face with the greatest edge similarity is taken (see fourth picture in Fig. 9).

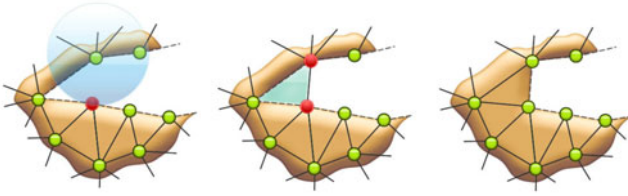


Fig. 8. Coalescing process at a mesh corner. On the left, the search process of a coalescing candidate. In the middle, one edge is created, on the right, the only face capable of being added is the corner face.

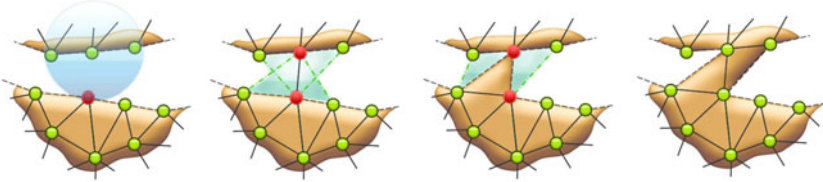


Fig. 9. Coalescing of two separate boundaries. In the second picture, the edge is determined, in the third, the triangle with smallest variance of edge lengths is added, in the fourth, another triangle must be added to avoid a needle eye.

C) Roughness Adaption. Up to now, the SGC are able to approximate an arbitrary sample set by a 2D mesh. What remains is an efficient local adaption of the mesh density in a way that areas with a strong curvature are modeled by a finer mesh resolution (see Fig. 10). This also relieves the influence of the sample density on the mesh granularity making the SGC less vulnerable to sampling artefacts like holes or regions which are not sampled with a uniform distribution.

Each time a vertex is adapted by a new sample the estimated normal \mathbf{n}_k at a neural vertex \mathbf{v}_k is calculated by the average of the normals at the surrounding faces. The curvature $c_k \in \mathbb{R}$ at a vertex is determined by

$$c_k = 1 - \frac{1}{|\mathcal{N}_k|} \sum_{\forall \mathbf{n} \in \mathcal{N}_k} \mathbf{n}_k \cdot \mathbf{n} \quad (1)$$

with the set \mathcal{N}_k containing the normals of the neighboring neural vertices of \mathbf{v}_k . Each time a neural vertex is selected as winner, its curvature value is calculated and a global curvature value \bar{c} is adjusted. Finally, the curvature dependent resource term r_k at \mathbf{v}_k is adapted through $r_k^{new} = r_k^{old} + \Delta r_k$, and

$$\Delta r_k = \begin{cases} 1, & \text{if } (c_k < \bar{c} + \sigma_{r_k}) \\ [c_k / (\bar{c} + \sigma_{r_k})] (1 - r_{min}) + r_{min} & \text{else,} \end{cases} \quad (2)$$

with the deviation σ_{r_k} of the resource term r_k , and a constant resource r_{min} that guarantees that the mesh does not completely vanish at plane regions with a very small curvature.

D) Discontinuity Cells. A sampled model that exposes discontinuities like edges is difficult to be approximated by the neural network mesh. Discontinuities are smoothed



Fig. 10. Roughness adaption correlates surface curvature with mesh density, details of the model are accentuated

out since the network tries to create a surface over them. This might be acceptable in many application areas since the approximation error is fairly small, but this effect is unfavorable in computer graphics since it is clearly visible. And even worse: edges are quite common in real world scenarios.

Therefore, we propose discontinuity neural vertices which, first, are only capable of moving in the direction of an object edge to represent them more properly, and second, the smoothing process is not applied to them.

Recognizing those vertices is accomplished as follows. The curvature values of those neighbors which have a distance of two connections from the vertex (the “second ring” of neighbors) are determined. Then the average δ_{ring} of the squared differences of consecutive curvature values on the ring is calculated.

If a curvature value clearly deviates from the average curvature value, it is assumed being a discontinuity vertex if the average of the neighbors’ (second ring) curvature gradient differs to a certain amount. Thus, a vertex \mathbf{v}_k is defined a discontinuity vertex if

$$(c_k > 2\sigma_{c_k}) \wedge (\forall c \in C_k : \delta_{ring} > 4\sigma_{c_k}^2) \tag{3}$$

with C_k the set of curvature values of the second ring of neighbors.

For approximating the edge normal the average of the normals of two of the neighboring vertices of \mathbf{v}_k are taken, either those with the highest curvature value, or those which are already marked as discontinuity vertex. Finally, the normal is mirrored if the edge angle lies above 180° indicated by the average of the surrounding vertex normals; in the first case it points in the direction of \mathbf{v}_k .

An Edge swap operation is applied if two connected discontinuity vertices grow into an edge, they nicely represent this edge by a triangle edge. But if the line is interrupted by a non-discontinuity vertex, a dent arises since this vertex is not placed on the edge. In this case an edge swap process is proposed which minimizes this effect.

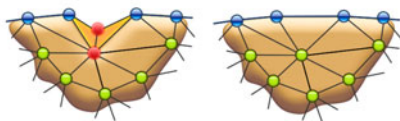


Fig. 11. A dent (left picture) on a sharp edge is solved (right picture) by an edge swap operation. Finally, connections of discontinuity vertices model object edges.

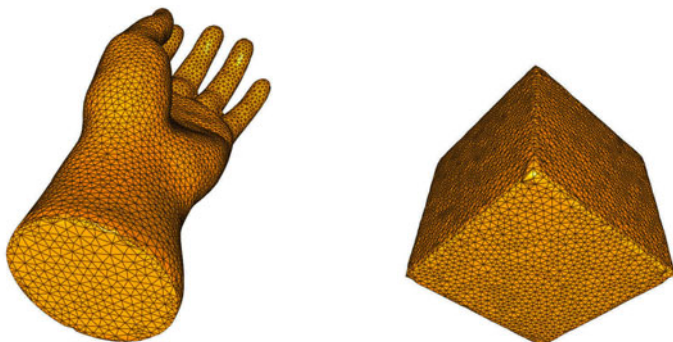


Fig. 12. Discontinuity vertices focus on edges. Edge swap operations let mesh edges map to object edges.

Each time a discontinuity vertex is moved towards a sample, the need for an edge swap operation will be determined by collecting the three consecutive faces with the most differing face normals. In case of a dent, the face in the middle is assumed to be the one which is misplaced and an edge swap operation is applied (see Fig. 11). Then, if the difference of the normals is now lower than before, edge swap is accepted, if not, the former structure is held.

Edge swap results in models where finally edges are represented by mesh boundaries (see Fig. 12).

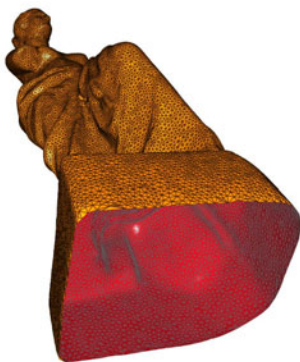


Fig. 13. Mesh boundary due to the missing bottom of the statue is represented exactly by boundary cells

E) Boundary Cells. Similar to discontinuity vertices which are capable of moving to object edges, boundary vertices are able to move to the outer border of a surface (see Fig. 13). They are recognized by being part of a triangle edge which is connected to one face only.

Then, these vertices are moved only into the direction of the boundary normal like described in section 2.2 for avoiding vertices just lying in the average of the surrounding samples but directly match the surface boundaries at their locations.

```

Adjust samples regarding roughness.
Calculate average curvature and deviations.
Recognize and sign discontinuity and curvature cells.
for all Boundary cells do
  if  $\exists$  coalescing candidate then
    Melt boundary.
    for all Weeding candidates do
      Weeding process.
    end for
  end if
end for
if Edge collapse operation triggered then
  Collapse edge.
  for all Weeding candidates do
    Weeding process
  end for
end if
if Vertex split operation triggered then
  Split vertex.
end if
    
```

Fig. 14. Outline of the complete SGC algorithm









								
Samples	36K	438K	544K	14,028K	5,000K	511K	38K	346K
Vertices	30K	100K	260K	320K	500K	10K	5K	346K
Time [m:s]	0:39	2:47	9:15	12:17	21:5	0:11	0:6	0:6
Quality	95.6	95.5	93.1	98.5	95.9	99.8	99	98.3
RMS/Size	4.7e-5	3.3e-5	1.7e-5	1.3e-5	2.7e-5	6.6e-5	15e-5	0.7e-5

Fig. 15. Results with sample sets from the *Stanford 3D Scanning Repository*. “Quality” means percentage of triangles which hold the Delaunay criterion. RMS/Size is the root of the squared distances between original point samples and the triangle mesh, divided by the diameter of the sample set.



Fig. 16. Upper lines: mesh training stages with number of vertices, lower lines, assorted pictures of reconstructed models.

2.3 Results

For the full algorithm of this approach see the pseudocode in Fig. 14. To keep it comprehensive, the outermost loop of the algorithm is neglected, and vertex split and edge collapse operations are triggered by counters.

Parameters which have been proven to be reliable for almost all sample sets we took for reconstruction are $\epsilon_{bm} = 0.1$, $\epsilon_{nb} = 0.08$, $r_{min} = 0.3$, $\epsilon_{acute} = 0.5$, $n_{degacute} = 4$, $k_{ins} = 100$, $k_{del} = 5$, $n_{degnb} = 1$, $n_{nastyacute} = 4$, $n_{nastyval} = 3$.

The following results have been produced on a *Dell*[®] Precision M6400 Notebook with *Intel*[®] Core 2 Extreme Quad Core QX9300 (2.53GHz, 1066MHz, 12MB) processor with 8GB 1066 MHz DDR3 Dual Channel RAM. The algorithm is not parallelized.

Visual results are exposed in Fig. 16. All pictures are drawn from an SGC mesh. Most models stem from the *Stanford 3D Scanning Repository*. Besides visual results, reconstruction with SGC comes up with impressive numbers compared to classical approaches, which are listed in the table in Fig. 15. It can be seen that mesh quality, i.e. the percentage of perfect triangles in the mesh lies at 96% at average. This is an outstanding but expected result, when using an approach from the field of unsupervised learning, since this guarantees an ideal representation of the underlying training sample distribution.

Further, the distance (RMS/object size) between samples and mesh surface is negligible low — far below 1% of the object size at average. This is even more pleasant, since usually the problem at edges generate big error terms. Also the computing times needed are very short, just few minutes in common cases.

All those measurements are far better than those from classical approaches, in so far as these are revealed in the specific papers. Our algorithm works very robustly. There are nearly no outliers visible in the mesh.

3 Conclusions

We presented smart growing cells as an expansion of general unsupervised learning. The novel core skill of a smart growing cell is the added intelligence — cells may not only adapt to the sample distribution but can also use application-focused aspects of the data they operate on. This extension is consistently injected into the standard unsupervised learning rule avoiding the extinction of its beneficial properties.

As a proof of concept a surface reconstruction algorithm was presented that overcomes the well-known problems of approaches that use the standard growing cells structures concept. Features like creases and corners are preserved, the triangle density relates to the curvature and arbitrary topologies are reconstructed. Our approach even outperforms classical surface reconstruction approaches.

For evaluation purposes, several sample sets were used which provide different reconstruction challenges (see table in Fig. 15). The average deviation of the mesh from the sample points is $2 \cdot 10^{-3} \%$ compared to the diameter of the object under consideration, and about 96% of the triangles fulfil the Delaunay criterion for triangle quality.

A further important feature of SGC is their robustness. The network is able to handle arbitrary topologies and billions of samples easily. It recognizes and solves discontinuities in the sample data and it is capable of adapting to varying sample distributions without the need for training from the scratch.

The network is able to match arbitrary surface structures like single objects, landscapes, or even separate objects with very complex topologies. The final triangulation is directly taken from the network structure. No additional triangulation or cleaning processes are required.

Future Work. We propose three directions for ongoing work on SGC. First, currently, the additional intelligence of a cell is purely defined on heuristics. A great improvement would be to use a separate neural network which detects and realizes cell behavior automatically. Second, in the area of surface reconstruction smart growing cells produced great results which is a proof of concept. To establish the flexibility of smart growing cells new applications cases will be realized. Third, the movement of a cell is independent from other cells. This offers great opportunities concerning the parallelisation of the presented approach. With a decent speed up real time application should be in reach.

References

1. Boissonnat, J.-D.: Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.* 3(4), 266–286 (1984)
2. Edelsbrunner, H., Mcke, E.P.: Three-dimensional alpha shapes (1994)
3. Fritzke, B.: Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks* 7, 1441–1460 (1993)
4. Fritzke, B.: A growing neural gas network learns topologies. In: Tesauro, G., Touretzky, D.S., Leen, T.K. (eds.) *Advances in Neural Information Processing Systems*, vol. 7, pp. 625–632. MIT Press, Cambridge (1995)
5. Hoffmann, M., Vrady, L.: Free-form surfaces for scattered data by neural networks. *Journal for Geometry and Graphics* 2, 1–6 (1998)
6. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J.A., Stuetzle, W.: Surface reconstruction from unorganized points. In: Thomas, J.J. (ed.) *SIGGRAPH*, pp. 71–78. ACM (1992)
7. Huang, Q.-X., Adams, B., Wand, M.: Bayesian surface reconstruction via iterative scan alignment to an optimized prototype. In: *SGP 2007: Proceedings of the fifth Eurographics Symposium on Geometry Processing*, pp. 213–223. Aire-la-Ville, Switzerland (2007); Eurographics Association
8. Ivriissimtzis, I.P., Jeong, W.-K., Seidel, H.-P.: Using growing cell structures for surface reconstruction. In: *SMI 2003: Proceedings of the Shape Modeling International 2003*, p. 78. IEEE Computer Society, Washington (2003)
9. Ivriissimtzis, I., Jeong, W.-K., Seidel, H.-P.: Neural meshes: Statistical learning methods in surface reconstruction. Technical Report MPI-I-2003-4-007, Max-Planck-Institut für Informatik, Saarbrücken (April 2003)
10. Ivriissimtzis, I., Lee, Y., Lee, S., Jeong, W.-K., Seidel, H.-P.: Neural mesh ensembles. In: *3DPVT 2004: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, pp. 308–315. IEEE Computer Society, Washington (2004)
11. Ivriissimtzis, I.P., Jeong, W.-K., Lee, S., Lee, Y., Seidel, H.-P.: Neural meshes: surface reconstruction with a learning algorithm. Research Report MPI-I-2004-4-005. Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany (October 2004)

12. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, SGP 2006, pp. 61–70. Aire-la-Ville, Switzerland (2006); Eurographics Association
13. Kohonen, T.: Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics* 43, 59–69 (1982)
14. Kolluri, R., Shewchuk, J.R., O’Brien, J.F.: Spectral surface reconstruction from noisy point clouds. In: SGP 2004: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 11–21. ACM, New York (2004)
15. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations, pp. 281–297 (1967)
16. Storvik, G.: Bayesian surface reconstruction from noisy images. In: *Interface 1996* (1996)
17. Taubin, G.: A signal processing approach to fair surface design. In: *SIGGRAPH*, pp. 351–358 (1995)
18. Vrády, L., Hoffmann, M., Kovcs, E.: Improved free-form modelling of scattered data by dynamic neural networks. *Journal for Geometry and Graphics* 3, 177–183 (1999)
19. Yoon, M., Lee, Y., Lee, S., Ivrişsimtzis, I., Seidel, H.-P.: Surface and normal ensembles for surface reconstruction. *Comput. Aided. Des.* 39(5), 408–420 (2007)
20. Yu, Y.: Surface reconstruction from unorganized points using self-organizing neural networks. In: *IEEE Visualization 1999, Conference Proceedings*, pp. 61–64 (1999)

Author Index

- Alba, Enrique 21
Al-Duwaish, Hussain N. 331
Alexandre, Frédéric 317
Alonso, César L. 49
Annuth, Hendrik 405

Behle, Christoph 345
Bernardino, Anabela Moreira 81
Bernardino, Eugénia Moreira 81
Bezdek, James C. 3
Bohn, Christian-A. 405
Boley, Harold 167
Borges, Cruz Enrique 49
Bye, Robin T. 131

Cadenas, J.M. 151
Casanova, Isidoro J. 183
Castro-Company, Francisco 261
Collier, Robert 99
Cona, Filippo 389

De, Arijit 197
Diaz, Elizaebeth 197
Du, Weichang 167

Echeandía, Marina de la Cruz 49

Fernandes, Carlos M. 115
Fidanova, Stefka 21
Frijns, Arjan J.H. 307

Garrido, M.C. 151
Gómez-Pulido, Juan Antonio 81
Guller, Dušan 211

Hashmi, Atif 373
Havens, Timothy C. 3

Henneges, Carsten 345
Hilderman, Robert 31

Kawashima, Masato 287
Kingma, Boris R.M. 307
Kroha, Petr 247
Kůrková, Věra 361

Lauschke, Marcus 247
Li, Yongtao 287
Lichtenbelt, Wouter D. van Marken 307
Lipasti, Mikko 373
Lopez-Garcia, P. 229

Marinov, Pencho 21
Martínez, R. 151
Massimini, Marcello 389
Matsumoto, Mitsuharu 277
Merelo, Juan Julián 115
Miyahara, Naoya 287
Montaña, José Luis 49
Morlino, Giuseppe 67
Muñoz-Hernández, S. 229

Nabeta, Kei-ichiro 287
Nara, Shigetoshi 287
Nicolai, Garrett 31

Palaniswami, Marimuthu 3
Puente, Alfonso Ortega de la 49

Rizvi, Syed Z. 331
Romaguera, Salvador 261
Rosa, Agostinho C. 115

- Rosanova, Mario 389
Rougier, Nicolas 317
Sánchez-Pérez, Juan Manuel 81
Saris, Wim H. 307
Takamura, Yuta 287
Taouali, Wahiba 317
Tirado, Pedro 261
Trianni, Vito 67
Tuci, Elio 67
Ursino, Mauro 389
van Steenhoven, Anton A. 307
Vega, T. Trigo de la 229
Vega-Rodríguez, Miguel Angel 81
Wineberg, Mark 99
Yamaguchi, Hitoshi 287
Yoshinaka, Ryosuke 287
Zavaglia, Melissa 389
Zell, Andreas 345
Zhao, Jidi 167