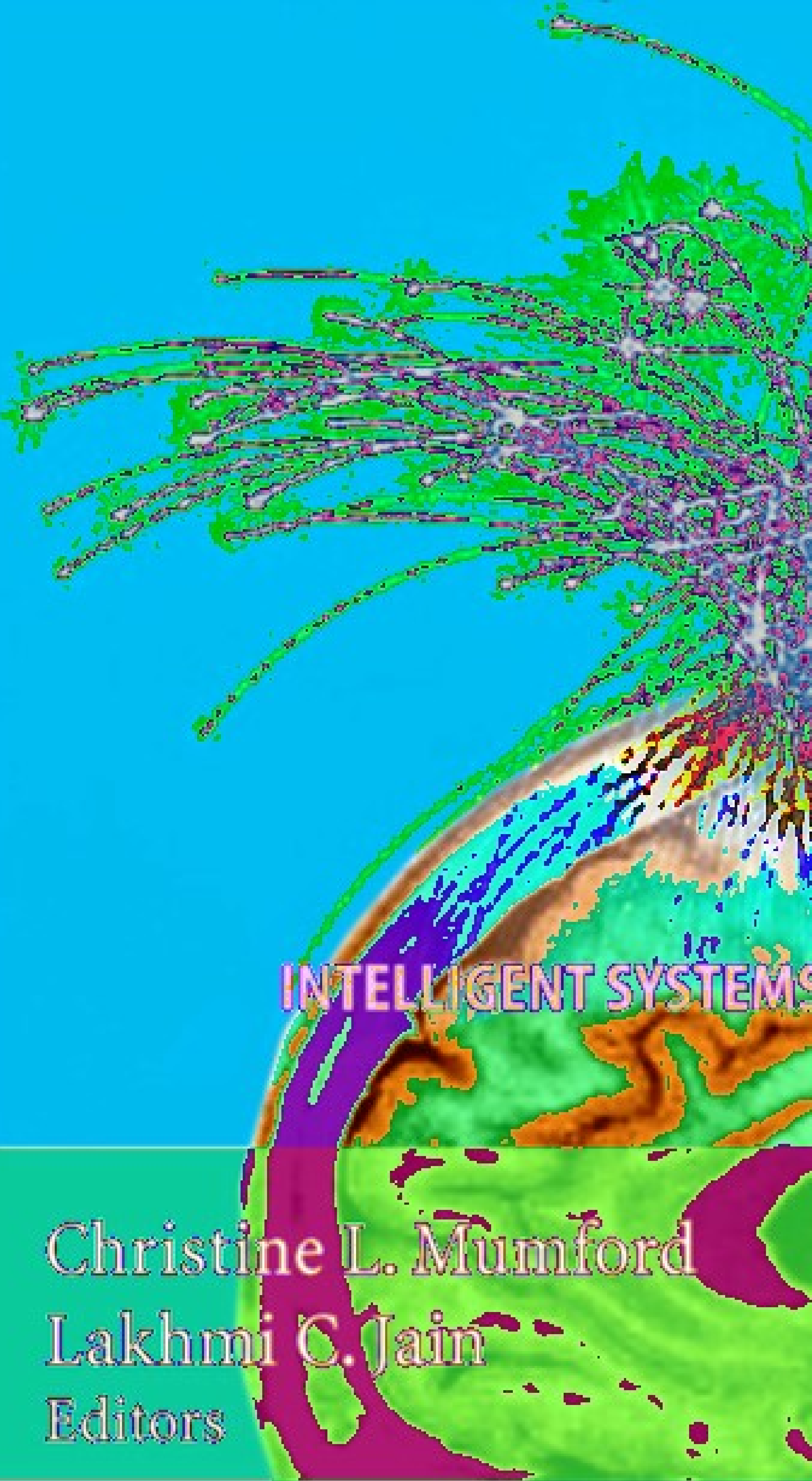


INTELLIGENT SYSTEMS

Christine L. Mumford
Lakhmi C. Jain
Editors

Christine L. Mumford and Lakhmi C. Jain (Eds.)

Computational Intelligence

Intelligent Systems Reference Library, Volume 1

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

Vol. 1. Christine L. Mumford and Lakhmi C. Jain (Eds.)
Computational Intelligence, 2009
ISBN 978-3-642-01798-8

Christine L. Mumford and Lakhmi C. Jain (Eds.)

Computational Intelligence

Collaboration, Fusion and Emergence

Springer

Dr. Christine L. Mumford
School of Computer Science
Cardiff University
5 The Parade, Roath
Cardiff, CF24 3AA
UK
E-mail: C.L.Mumford@cs.cardiff.ac.uk

Prof. Lakhmi C. Jain
University of South Australia
Adelaide
Mawson Lakes Campus
South Australia
Australia
E-mail: Lakhmi.jain@unisa.edu.au

ISBN 978-3-642-01798-8 e-ISBN 978-3-642-01799-5

DOI 10.1007/978-3-642-01799-5

Intelligent Systems Reference Library ISSN 1868-4394

Library of Congress Control Number: Applied for

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Dedicated to the chapter authors.

Editors

Dr Christine Mumford is a Senior Lecturer at Cardiff University in the School of Computer Science. She is a Senior Member of the IEEE. Her research interests include evolutionary computing and other metaheuristics, multi-objective optimization, and applications focused on combinatorial optimization, particularly vehicle routing, logistic network design, scheduling, timetabling and cutting and packing.

Professor Lakhmi C. Jain is a Director/Founder of the Knowledge-Based Intelligent Engineering Systems (KES) Centre, located in the University of South Australia. He is a fellow of the Institution of Engineers Australia.

His interests focus on the artificial intelligence paradigms and their applications in complex systems, art-science fusion, virtual systems, e-education, e-healthcare, unmanned air vehicles and intelligent agents.

Preface

This book is about *synergy* in computational intelligence (CI). It is a collection of chapters that covers a rich and diverse variety of computer-based techniques, all involving some aspect of computational intelligence, but each one taking a somewhat pragmatic view. Many complex problems in the real world require the application of some form of what we loosely call "intelligence" for their solution. Few can be solved by the naive application of a single technique, however good it is. Authors in this collection recognize the limitations of individual paradigms, and propose some practical and novel ways in which different CI techniques can be combined with each other, or with more traditional computational techniques, to produce powerful problem-solving environments which exhibit synergy, i.e., systems in which *the whole is greater than the sum of the parts*[1].

Computational intelligence is a relatively new term, and there is some disagreement as to its precise definition. Some practitioners limit its scope to schemes involving evolutionary algorithms, neural networks, fuzzy logic, or hybrids of these. For others, the definition is a little more flexible, and will include paradigms such as Bayesian belief networks, multi-agent systems, case-based reasoning and so on. Generally, the term has a similar meaning to the well-known phrase "Artificial Intelligence" (AI), although CI is perceived more as a "bottom up" approach from which intelligent behaviour can emerge, whereas AI tends to be studied from the "top down", and derive from pondering upon the "meaning of intelligence". (These and other key issues will be discussed in more detail in Chapter 1.) In this book we will take a relatively broad view of CI.

Common themes to be found in the various chapters of this collection include the following: *fusion*, *collaboration* and *emergence. Fusion* describes the hybridization of two or more techniques, at least one of which will involve CI. *Fused* techniques need to *Collaborate* in order to "work together" harmoniously on the required application. Distributed CI techniques, such as

[1] This phrase is attributed originally to Aristotle.

neural networks and multi-agent systems are also collaborative in their nature, and all such systems require *effective communication. Emergence* refers to the phenomenon that complex behaviour can emerge from collaboration between simple processing elements - indeed, many would say that this is the key to success. The twenty two chapters have been grouped into nine parts (see Table 1):

I. Introduction

II. Fusing evolutionary algorithms and fuzzy logic

III. Adaptive solution schemes

IV. Multi-agent systems

V. Computer vision

VI. Communication for CI systems

VII. Artificial immune systems

VIII. Parallel evolutionary algorithms

IX. CI for clustering and classification

This book is aimed at a broad audience: graduate students, researchers, engineers, and computer scientists. The idea is to try to motivate the reader to explore cutting-edge challenges that may sit on the periphery of their present areas of interest. Most chapters include a gentle introduction to the topics they address, and thus should prove interesting to the relative beginner as well as to the more experienced reader. All chapters provide suggestions for background and further reading.

Acknowledgments

We are grateful to the authors for their wonderful contribution, and to the reviewers for their excellent comments which helped to improve the quality of chapters. Thanks are also due to Springer-Verlag for their excellent support during the preparation of the manuscript.

March 2009

Christine Mumford
Cardiff University, UK
Lakhmi C. Jain
University of South Australia, Australia

Table 1 The chapters and themes of the book

Theme	Chapter
Introduction	1: Synergy in Computational Intelligence
	2: Computational Intelligence: The Legacy of Alan Turing and John von Neumann
Evolutionary Algorithms and Fuzzy Logic	3: Multiobjective Evolutionary Algorithms for the Electric Power Dispatch Problem
	4: Fuzzy Evolutionary Algorithms and Genetic Fuzzy Systems: A Positive Collaboration Between Evolutionary Algorithms and Fuzzy Systems
	5: Multiobjective Genetic Fuzzy Systems
Adaptive Solution Schemes	6: Exploring Hyper-Heuristic Methodologies with Genetic Programming
	7: Adaptive Constraint Satisfaction: The Quickest First Principle
Multi-Agent Systems	8: Collaborative Computational Intelligence in Economics
	9: IMMUNE: A Collaborating Environment for Complex System Design
	10: Bayesian Learning for Cooperation in Multi-Agent Systems
	11: Collaborative Agents for Complex Problem Solving
Computer vision	12: Predicting Trait Impressions of Faces Using Classifier Ensembles
	13: The Analysis of Crowd Dynamics: From Observations to Modelling
Communication for CI	14: Computational Intelligence for the Collaborative Identification of Distributed Systems
	15: Collaboration at the Basis of Sharing Focused Information: The Opportunistic Networks
Artificial Immune Systems	16: Exploiting Collaborations in the Immune System: The Future of Artificial Immune Systems
Parallel EAs	17: Evolutionary Computation: Centralized, Parallel or Collaborative
Clustering and Classification	18: Fuzzy Clustering of Likelihood Curves for Finding Interesting Patterns in Expression Profiles
	19: A Hybrid Rule Induction/Likelihood Ratio-Based Approach for Predicting Protein-Protein Interactions
	20: Improvements in Flock-based Collaborative Clustering Algorithms
	21: Combining Statistics and Case-Based Reasoning for Medical Research
	22: Collaborative and Experience-Consistent Schemes of System Modelling in Computational Intelligence

Contents

Part I: Introduction

Synergy in Computational Intelligence 3
Christine L. Mumford

Computational Intelligence: The Legacy of Alan Turing and John von Neumann 23
Heinz Mühlenbein

Part II: Fusing Evolutionary Algorithms and Fuzzy Logic

Multiobjective Evolutionary Algorithms for Electric Power Dispatch Problem 47
Mohammad A. Abido

Fuzzy Evolutionary Algorithms and Genetic Fuzzy Systems: A Positive Collaboration between Evolutionary Algorithms and Fuzzy Systems 83
F. Herrera, M. Lozano

Multiobjective Genetic Fuzzy Systems 131
Hisao Ishibuchi, Yusuke Nojima

Part III: Adaptive Solution Schemes

Exploring Hyper-heuristic Methodologies with Genetic Programming 177
Edmund K. Burke, Mathew R. Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, John R. Woodward

Adaptive Constraint Satisfaction: The Quickest First Principle 203
James E. Borrett, Edward P.K. Tsang

Part IV: Multi-agent Systems

Collaborative Computational Intelligence in Economics 233
Shu-Heng Chen

IMMUNE: A Collaborating Environment for Complex System Design 275
Mahmoud Efatmaneshnik, Carl Reidsema

Bayesian Learning for Cooperation in Multi-Agent Systems 321
Mair Allen-Williams, Nicholas R. Jennings

Collaborative Agents for Complex Problems Solving 361
Minjie Zhang, Quan Bai, Fenghui Ren, John Fulcher

Part V: Computer Vision

Predicting Trait Impressions of Faces Using Classifier Ensembles 403
Sheryl Brahnam, Loris Nanni

The Analysis of Crowd Dynamics: From Observations to Modelling 441
B. Zhan, P. Remagnino, D.N. Monekosso, S. Velastin

Part VI: Communications for CI Systems

Computational Intelligence for the Collaborative Identification of Distributed Systems 475
Giorgio Biagetti, Paolo Crippa, Francesco Gianfelici, Claudio Turchetti

Collaboration at the Basis of Sharing Focused Information: The Opportunistic Networks 501
Bruno Apolloni, Guglielmo Apolloni, Simone Bassis, Gian Luca Galliani, Gianpaolo Rossi

Part VII: Artificial Immune Systems

Exploiting Collaborations in the Immune System: The Future of Artificial Immune Systems 527
Emma Hart, Chris McEwan, Despina Davoudani

Part VIII: Parallel Evolutionary Algorithms

Evolutionary Computation: Centralized, Parallel or Collaborative 561
Heinz Mühlenbein

Part IX: CI for Clustering and Classification

Fuzzy Clustering of Likelihood Curves for Finding Interesting Patterns in Expression Profiles 599
Claudia Hundertmark, Lothar Jänsch, Frank Klawonn

A Hybrid Rule-Induction/Likelihood-Ratio Based Approach for Predicting Protein-Protein Interactions 623
Mudassar Iqbal, Alex A. Freitas, Colin G. Johnson

Improvements in Flock-Based Collaborative Clustering Algorithms 639
Esin Saka, Olfa Nasraoui

Combining Statistics and Case-Based Reasoning for Medical Research 673
Rainer Schmidt, Olga Vorobieva

Collaborative and Experience-Consistent Schemes of System Modelling in Computational Intelligence 697
Witold Pedrycz

Index 725

Part I
Introduction

Synergy in Computational Intelligence

Christine L. Mumford

Abstract. This chapter introduces the book. It begins with a historical perspective on Computational Intelligence (CI), and discusses its relationship with the longer established term "Artificial Intelligence" (AI). The chapter then gives a brief overview of the main CI techniques, and concludes with short summaries of all the chapters in the book.

1 Introduction

In the early days of information technology computers were large, expensive and the property of the few government organizations, academic institutions and big businesses who could afford them. Centralized operating systems were developed and two classes of computer systems evolved: one for scientific computing and engineering, specializing in "number crunching" and the other for business computing focussing on data processing activities such as stock control and computerized customer accounts. Today computing devices are small and cheap, and pervade our every day lives. It is therefore not surprising that the style of software required for the twenty-first century is very different from that needed to run operations on the large mainframe computers of the past. It is in this climate that the field of "Artificial Intelligence" (AI) has given way to the newer study of "Computational Intelligence" (CI)[1]. AI grew out of attempts to emulate the human brain on mainframe computers, while CI is more pragmatic and relies on distributed computation, communication and emergence. CI is well suited to today's modern ubiquitous computing devices.

This book is about practical computational intelligence. It covers many techniques and applications, and focuses on novel ways of combining different CI

Christine L. Mumford
Cardiff University, School of Computer Science, 5 The Parade, Cardiff, CF24 3AA,
United Kingdom
e-mail: C.L.Mumford@cs.cardiff.ac.uk

[1] Terms with very similar meanings have also emerged in the recent literature, such as "soft computing" and "natural computing".

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 3–21.
springerlink.com

techniques together, or hybridizing CI techniques with traditional computational techniques. Recognizing the need for pragmatism, authors in this collection propose some new and exciting problem-solving frameworks. The key themes emphasized in the book title are *collaboration*,*fusion* and *emergence*. *Fusion* refers to the amalgamation of CI techniques with each other or with more traditional computational methods. *Collaboration* involves effective communication and is essential, if the above mentioned "fused" techniques are to work harmoniously together. Finally, *emergence* can be viewed as a central goal of CI, asserting that complex behaviour can emerge from collaboration between simple processing elements. An essential ingredient of a CI system exhibiting emergent behaviour is *synergy* in which *the whole is greater than the sum of the parts*.

The remainder of this chapter is structured as follows. It will begin with some discussion on the origins of Computational Intelligence, and examine its relationships with Artificial Intelligence. This will be followed by a brief survey of some of the key CI paradigms. The chapter will conclude with a brief overview of the rest of the book.

2 The Birth of Computational Intelligence

The origin of the term "Computational Intelligence" (CI) has been widely attributed to Bezdek [1, 2]. Defining a new field devoted to computer-based intelligence can be viewed as a timely attempt to escape from some of the difficult issues and bad publicity associated with the longer established field of Artificial Intelligence (AI). Although AI and CI have much in common, the emphasis is subtly different. CI concentrates on practical application, self organization and the emergence of complex behaviour from simple components, while AI aims to build intelligent systems based on ideas of how the human brain works. John McCarthy originally coined the term "Artificial Intelligence" in 1955, in advance of a month long brainstorming conference held in Dartmouth College in the following year. The proposal for the Dartmouth conference [15] makes interesting reading. The introduction is reproduced below.

> We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

The document goes on to discuss the "various aspects of the artificial intelligence problem" in more detail, including computers and computer programming, natural language processing, neural networks, the theory of computation, the need for automatic self-improvement, and aspects of abstraction and creativity. Most of these topics remain active research issues to this day. However, the assumption that human intelligence can be simulated by machine was perhaps a little overoptimistic. Indeed, it is one of the "big questions" remaining in computer science.

The two decades following the 1956 conference saw many high profile AI research projects, for example, the development of the LISP and PROLOG programming languages, the SHRDLU "microworlds" project, and the first expert systems (see standard texts on AI, such as [20, 21], for more information). Although few could argue that these projects had produced some highly successful results, and useful applications, there was, nevertheless, a general feeling of disappointment at the time, that the AI community had in some sense "failed to deliver". This perception was effectively articulated in a report to the British Science Research Council by the British academic James Lighthill in 1973 [14]:

> In no part of the field have discoveries made so far produced the major impact that was then promised.

In essence, the so-called "Lighthill Report" stated that AI researchers had failed to address the issue of the combinatorial explosion, i.e., AI techniques may work on small problem domains, but the techniques do not scale up well to solve more realistic problems. Following this very pessimistic view, the Science Research Council slashed funding for AI projects in the UK. Although a rather more optimistic view prevailed in much of the rest of the world, and major new investments continued throughout the 1980s (e.g., CYC in the USA [13], and the Fifth Generation Computer Systems project in Japan [6]). AI was becoming an increasingly fragmented study, consisting of many disciplines, such as reasoning, knowledge engineering, planning, learning, communication, perception, and so on. Despite the many successes that had been achieved using expert systems, logic programming, neural networks etc., it was blatantly obvious that the dream of properly emulating human intelligence had never come close to being realized. It was time to perhaps "move on" and capitalize on the substantial achievements provided by some of the "offshoots" of AI, and leave behind the very negative image that had become so closely associated with the term "AI" itself, not so much because AI had failed per se, but rather because of the over-inflated expectations that had become intrinsically tied up with the notion of it.

Bezdek's view of CI was as a system that exhibited some form of "intelligence", yet dealt with numerical (low level) data, as opposed to "knowledge", and in this sense differed from traditional Artificial Intelligence. Nevertheless, the view of Bezdek was very much focussed towards his personal research interests of pattern recognition and neural networks. In the following years the term "CI" became firmly established when it was adopted by the IEEE (the Institute of Electrical and Electronic Engineers), and in 2004 the Computational Intelligence Society (CIS) was established (as a name change from the Neural Network Society). The slogan of the IEEE CIS is "mimicking nature for problem solving", and its scope is stated as:

> The Field of Interest of the Society shall be the theory, design, application, and development of biologically and linguistically motivated computational paradigms emphasizing neural networks, connectionist systems, genetic algorithms, evolutionary

> programming, fuzzy systems, and hybrid intelligent systems in which these paradigms are contained.

Artificial intelligence brings its connotations of “intelligence”, which can be distracting. One can get sidetracked into pondering the meaning of intelligence, rather than asking more useful questions, about self-organization, and emergence of complex systems from simple components, for example. A useful definition taken from the Computer Science web site of Amsterdam University (http://www.cs.vu.nl/ci/) emphasizes the “bottom up” nature of CI:

> Enclosed in the name computational intelligence is a ‘message’, according to scientific folklore it is chosen to indicate the link to and the difference with artificial intelligence. While some techniques within computational intelligence are often counted as artificial intelligence techniques (e.g., genetic algorithms, or neural networks) there is a clear difference between these techniques and traditional, logic based artificial intelligence techniques. In general, typical artificial intelligence techniques are top-to-bottom, where the structure of models, solutions, etc. is imposed from above. Computational intelligence techniques are generally bottom-up, where order and structure emerges from an unstructured beginning.

Some interesting further discussions on the birth of AI and CI, and on some of the important philosophical issues on the essence of intelligence can be found in Chapter 2 of this book.

3 The Main CI Techniques

In this section we will look briefly at the following key CI paradigms: Evolutionary Algorithms, Neural Networks, Fuzzy Systems and Multi-Agent Systems. This will be followed by a short summary covering some other important techniques included by various authors in this collection.

3.1 Evolutionary Algorithms

Evolutionary algorithms (EAs) comprise a class of techniques inspired by evolution and natural selection. The best known EAs are undoubtedly the *genetic algorithms* (GAs) developed by John Holland [9] in the 1960’s and 70’s. Contemporaries of Holland independently developed some similar techniques however, for example of Rechenberg [19] introduced *evolution strategies* (ES) and Fogel, Owen and Walsh [7] developed *evolutionary programming* (EP). Since these early days, interest in evolutionary-inspired algorithms has grown extensively, and many new variations have appeared, often very different from the original models conceived by Holland, Rechenberg or Fogel. For example, in the early 1990s, John Koza proposed *genetic programming* [11]: an evolutionary style technique for evolving effective computer programs, mostly using the LISP programming language (see also Chapter 6). Other popular paradigms to have been derived from the more generic approach include artificial life [12], evolvable hardware [8], ant systems [4] and particle swarms [10] (Chapter 20), to name but a few. Artificial Immune Systems (Chapter 16) have also become a popular topic for research in recent years, drawing analogies with some of the ingenious problem-solving mechanisms observed in natural immune systems and applying them to a broad range of real-world problems. In addition, there are many examples of hybrid (or memetic) approaches where problem specific heuristics, or other techniques such as neural networks, fuzzy systems, or simulated annealing, have been incorporated into a GA framework. Thus, due to the growth in popularity of search and optimization techniques inspired by natural evolution during the last few decades, it is now common practice to refer to the field as *evolutionary computing* and to the various techniques as *evolutionary algorithms*. In addition, evolutionary techniques for simultaneously optimizing several objectives have recently become popular. These approaches, collectively known as multi-objective evolutionary algorithms [3] are very effective at balancing the frequently conflicting objectives to produce excellent trade-off solutions, from which a human decision maker can make an informed choice. Chapters 3 and 5 deal with multi-objective optimization problems.

Parallel evolutionary algorithms are discussed in Chapter 17. The analogy with natural population structures and their geographical distributions make parallel implementations highly desirable, to speed up processing and to facilitate complex emergent behaviour from simple components within the distributed populations.

Given the range of EAs mentioned above, it is not perhaps surprising that there is no rigorous definition of the term “evolutionary algorithm” that everyone working in the field would agree on. There are, however, certain elements that the more generic types of EA tend to have in common:

1. a population of chromosomes encoding candidate solutions to the problem in hand,
2. a mechanism for reproduction,
3. selection according to a fitness, and
4. genetic operators.

Figure 1 gives an outline of a generic EA. The process is initialized with a starting population of candidate solutions. The initial population is frequently generated by some random process, but may be produced by constructive heuristic algorithms, or by other methods. Once generated, the candidate solutions are evaluated to establish the quality of each solution, and based on this quantity, a fitness value will be computed, in such a way that better quality solutions will be assigned higher values for their fitness. Individuals will next be selected from the population to form the parents of the next generation, and these will be duplicated and placed in a mating pool. The selection process is frequently biased, so that fitter individuals are more likely to be chosen than their less fit counterparts. Genetic operators are then applied to the individuals in the mating pool. The idea is to introduce new variation, without which no improvement is possible. Recombination (also known as crossover) is achieved by combining elements of two parents to form new offspring. Mutation, on the other hand, involves very small random changes made to solutions. The final stage in the

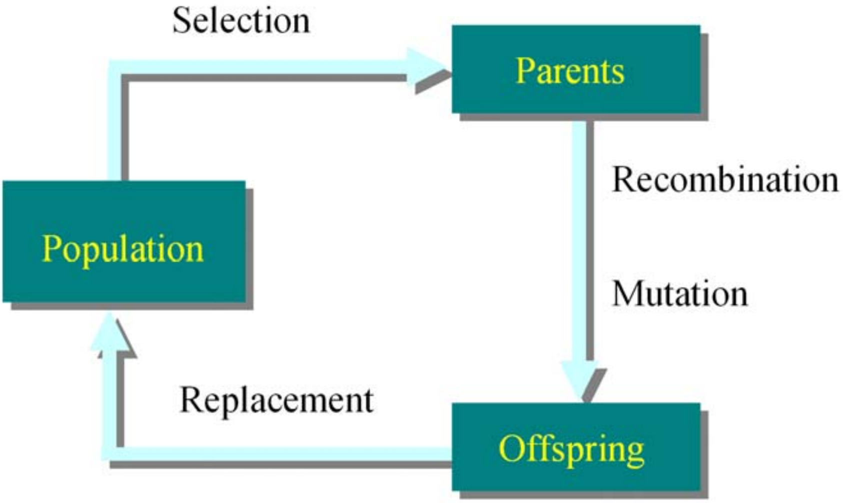


Fig. 1 The Evolutionary Cycle

cycle requires the population is updated with new individuals. Depending on the style of the EA, this may involve replacing the parent population in its entirety, or partial replacement is favoured by some researchers - perhaps replacing the poorest 10 % of the population by the best offspring, for example. A good general text on evolutionary algorithms is Eiben and Smith [5].

3.2 Neural Networks

Artificial Neural Networks (ANNs) are inspired by biological nervous systems, and emulate a simple “brain”. They consist of large numbers of highly interconnected processing elements (neurons) working together and learning from experience. ANNs are specially configured for each application, and typical uses include pattern recognition and data classification. In a biological nervous systems, learning involves making adjustments to the synaptic connections between the neurons. In a similar way for ANNs, learning is accomplished through the adjustment of weights by application of some “learning rule” to the connections between the artificial neurons or nodes. Learning rules typically attempt to reinforce connections that contribute to a “correct output”, and penalize connections that produce incorrect results. There are three main classes of ANN, distinguished by their different learning processes: 1) supervised learning, 2) unsupervised learning, and 3) reinforcement learning. With supervised learning a training stage uses a set of test data and a teacher to score the performance of the ANN, then adjusts the connection weights in an effort to improve the performance to better match the actual output to the predicted output. The most widely known supervised learning ANNs are the backpropagation nets. ANNs that use unsupervised learning do not have a training

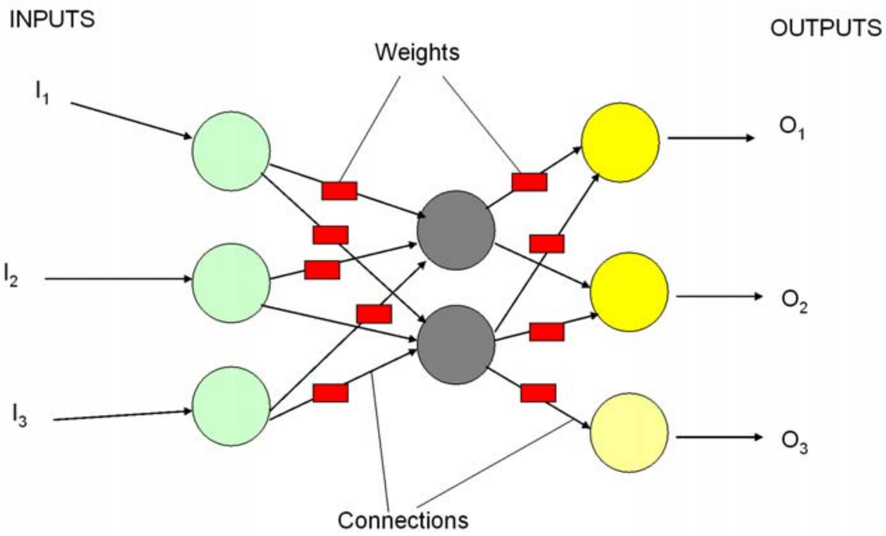


Fig. 2 A Neural Network with One Hidden Layer

stage, and these are frequently referred to as “self organizing networks”. Kohonen nets are the best known example of this type. In reinforcement learning data is not usually available. Instead the aim is to discover a policy for selecting actions that minimize some measure of long-term cost. A schematic neural network is illustrated in Figure 2. For more details on ANN see Mehrotra, Mohan, and Ranka [16]. Chapters 12, 13 and 22 all utilize neural networks, in one form or another.

3.3 Fuzzy Systems

Fuzzy logic was first proposed by Lotfi A. Zadeh of the University of California at Berkeley in a 1965 paper [23]. It is a modification of boolean (or crisp) logic which allows approximate and common sense reasoning in the absence of “true” or “false” certainty. In crisp logic, set membership is “all or nothing”. In contrast, fuzzy logic allows partial membership of sets, known as *fuzzy sets*, and forms the basis of *fuzzy systems*. Fuzzy Systems can deal with partial truth and incomplete data, and are capable of producing accurate models of how systems will behave in the real world, particulary when appropriate conventional system models are not available. Instead of supplying equations for a mathematical model, for example, a designer will need to produce appropriate fuzzy rules to describe the system he/she wishes to implement. The system operates when inputs are applied to the rules consisting of the current values of appropriate membership functions. Once activated, each rule will fire and produce an output, which will also be a partial truth value. In the final stage, the outputs from all the rules are combined, in some way, and converted into a single crisp output value. In summary, a fuzzy system consists of the following:

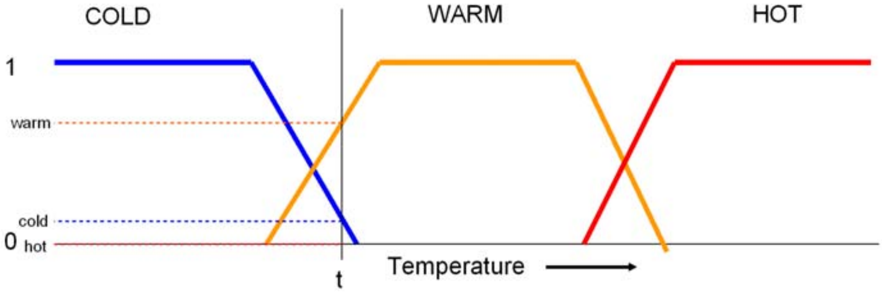


Fig. 3 A Fuzzy Temperature Control System

- a set of inputs
- a fuzzification system, for transforming the raw inputs into grades of memberships of fuzzy sets
- a set of fuzzy rules
- an inference system - to activate the rules and produce their outputs
- a defuzzification system - to produce one or more final crisp outputs

We will now look at a simplistic fuzzy system: a fuzzy controller for room temperature.

The fuzzy set membership diagram in Figure 3 characterizes three functions, identifiable as subranges of temperature: cold, warm and hot. Suppose we wish to keep a room at a comfortable temperature (warm) by building a control system to adjust a room heater. We can see in Figure 3 how each function maps the same temperature value to a truth value in the 0 to 1 range, so that any point on that scale has three "truth values", one for each of the three functions. It is these truth values that are used to determine how the room temperature should be controlled. The vertical line in the diagram represents a particular temperature, t. At this temperature it is easy to observe the degree of membership to "hot" (red) is zero, this temperature may be interpreted as "not hot". Membership of "warm" is about 0.7, and this may be described as "fairly warm". Similarly, examining membership of the "cold" function gives a value of about 0.15, which may describe it as "slightly cold". Adjectives such as "fairly" and "slightly", used to modify functions are referred to as "hedges", and can be a useful way to specify subregions of the functions to which they are applied.

To operate our fuzzy temperature control system, we require a number of fuzzy IF-THEN rules, in the form of "IF variable IS property THEN action". For example, an extremely simple temperature regulator that uses a heater might look like this:

1. IF temperature IS cold THEN turn heater to high
2. IF temperature IS warm THEN do nothing
3. IF temperature IS hot THEN turn off heater

Notice there is no "ELSE". All of the rules are evaluated, because the temperature will belong to all three sets (cold, warm and hot) at the same time, but to different degrees. At temperature t in Figure 3, for example, $M(cold) = 0.15, M(warm) = 0.7$ and $M(hot) = 0$.

Obviously, the greater the truth value of “cold”, the higher the truth value of “turn the heater to high”, although this does not necessarily mean that the output itself will be set to “high”, since this is only one rule among many. In our example, the partial truth inputs for “cold”, “warm” and “hot” will in turn produce partial truth values for the outputs “turn the heater to high”, “do nothing” and “turn the heater off”. The simplest way to produce a single crisp instruction, is to select the output with the maximum value (which will probably map to “do nothing” in the case of our temperature t). A more sophisticated method involves finding the centroid of all the outputs. This methods locates the “centre of mass” of the combined membership function curves.

More complex rules can be built for fuzzy systems, using AND, OR, and NOT operators. These are the counterparts of the familiar crisp logic operators, and they are usually defined (respectively) as the minimum, maximum, and complement. So, for the fuzzy variables x and y:

NOT x = (1 - truth(x))
x AND y = minimum(truth(x), truth(y))
x OR y = maximum(truth(x), truth(y))

Clearly, the simple temperature controller described above is for illustration only, and practical fuzzy systems will typically be made up from many more rules - perhaps hundreds or even thousands. In these more sophisticated systems, it is likely that the fuzzy rule set will be less “flat”, and form more of a hierarchy, so that the outputs of some rules provide inputs to others. Systems with large rule sets will probably require more sophisticated inference systems to ensure the efficient processing of the rules, in a reasonable order.

To complete this section, it is worth mentioning a variation of fuzzy sets called *rough sets*. Rough Set Theory was introduced in the early 1980s by Zdzislaw Pawlak [18]. The basic idea is to take concepts and decision values, and create rules for upper and lower boundary approximations of the set. With these rules, a new object can easily be classified into one of the regions. Rough sets are especially helpful in dealing with vagueness and uncertainty in decision situations, and for estimating missing data. Uses include data mining, stock market prediction and financial data analysis, machine learning and pattern recognition.

For further reading on fuzzy systems [17] is a good introductory text. Also Chapter 4 in the present book, provides a good background to many of the important concepts, and chapters 3, 5, 18, and 22 also cover aspects of fuzzy systems.

3.4 Multi-Agent Systems

A multi-agent system (MAS) is a system composed of many interacting intelligent agents; each one is in itself simple and apparently acts only in its own interest, yet by collaborating and/or competing with each other agents, an MAS can be used to solve problems which would entirely defeat an individual agent or a monolithic system. MAS can exhibit self-organization and complex behaviour can emerge. Example applications include financial forecasting and online trading (see Chapter 8) and disaster response (see Chapter 10).

The agents in a multi-agent system have several important characteristics [22]:

- Autonomy: the agents are at least partially autonomous
- Local views: no agent has a full global view of the system
- Decentralization: there is no one controlling agent
- Typically multi-agent systems research refers to software agents. However, the agents in a multi-agent system could equally well involve robots, humans or human teams. A multi-agent system may contain combined human-software agent teams (see Chapter 8).

Generally, multi-agent systems are flexible and they are easily maintained or modified without the need for drastic rewriting or restructuring. MAS also tend to be robust and recover easily from a breakdown, due to built in duplication and redundancy of components. Chapters 8, 9, 10, 11 and 20 all deal explicitly with multi-agent systems.

3.5 Other Techniques Covered in the Book

Besides the main methods outlined above, a number of other CI techniques have been used by various authors in this text, including rule induction (Chapter 19), Bayesian Learning (Chapter 10), Likelihood Ratios (Chapters 18 and 19), Case-Based Reasoning (Chapter 21), Collaborative Clustering (Chapter 22), Blackboard Database Systems (Chapter 9), and Hyper-Heuristics (Chapter 6). Among the "traditional techniques" used in partnership with the CI methods, statistical methods are used in Chapters 13 and 21, and computer vision techniques in Chapters 12 and 13. Effective communications are essential for agent-based systems and all distributed CI techniques. These important issues are addressed in Chapters 14 and 15.

4 Chapters Included in This Book

This book is divided into nine parts:

Introduction
Fusing evolutionary algorithms and fuzzy logic
Adaptive solution schemes
Multi-agent systems
Computer vision
Communication for CI systems
Artificial immune systems
Parallel evolutionary algorithms
CI for clustering and classification

4.1 Part I: Introduction

This Part covers some of the history of computational intelligence, and sets the scene for the rest of the book.

Chapter 1: Synergy in Computational Intelligence

The present chapter, by Christine Mumford, introduces the book and begins Part I. It begins with a brief history of Artificial Intelligence and discusses the origins of the term “Computational Intelligence”. Then follows an introduction to the main Computational Intelligence paradigms used by the various authors in the book; and finally, the chapter concludes with short summaries of all the individual chapters.

Chapter 2: Computational Intelligence: The Legacy of Alan Turing and John von Neumann

In this thought-provoking chapter, Heinz Mühlenbein recalls the fundamental research questions of computational intelligence, and explains how many of these issues remain unresolved to this day. In recent years, it has become fashionable to subdivide computational intelligence into many fields e.g. evolutionary computation, neural networks, fuzzy logic. This was not always the case. This chapter recalls the broader issues and reviews the seminal research of Alan Turing and John von Neumann in detail. The author discusses the many areas of computational intelligence that need to come together, if we are to create automata with human-like intelligence.

4.2 Part II: Fusing Evolutionary Algorithms and Fuzzy Logic

These three chapters cover some useful ways to combine evolutionary algorithms with fuzzy systems.

Chapter 3: Multiobjective Evolutionary Algorithms for the Electric Power Dispatch Problem

The main objective of the electric power dispatch problem is to schedule the available generating units to meet the load demand at minimum cost, while satisfying all constraints. However, thermal plants are a major source of atmospheric pollution. Recently the pollution minimization problem has attracted a lot of attention as the public demand clean air. Mohammad Abido explores the use of evolutionary multiobjective optimization to minimize cost and pollution, simultaneously. Furthermore, he uses fuzzy set theory to select the “best” compromise solution from the trade-off solution set.

Chapter 4:Fuzzy Evolutionary Algorithms and Genetic Fuzzy Systems: A Positive Collaboration Between Evolutionary Algorithms and Fuzzy Systems

Two alternative ways of integrating fuzzy logic and evolutionary algorithms are discussed in detail by F. Herrera, M. Lozano in this chapter. The first one, called a *genetic fuzzy system (GFS)* consists of a fuzzy rule based system (FRBS) augmented by a learning process based on evolutionary algorithms. In the second approach, fuzzy tools and fuzzy logic-based techniques are used for modeling different evolutionary algorithm components and also for adapting evolutionary algorithm control parameters, with the goal of improving performance. The evolutionary algorithms resulting from the second type of integration are called *fuzzy evolutionary algorithms*. This chapter includes some excellent introductory material on fuzzy logic, as well as a summary of state-of-the-art with respect to genetic fuzzy systems and fuzzy evolutionary algorithms. The potential benefits derived from the synergy between evolutionary algorithms and fuzzy logic are made clear.

Chapter 5: Multiobjective Genetic Fuzzy Systems

Hisao Ishibuchi and Yusuke Nojima describe the two conflicting goals in the design of fuzzy rule-based systems: one is accuracy maximization, and the other is complexity minimization. Generally, complex rules and large rule sets promote accuracy, and smaller rule sets with simple rules reduce complexity. The authors discuss the trade-off relation between these two goals, i.e., that improving the accuracy of a rule set will simultaneously increase its complexity. This chapter explains how various studies in multiobjective genetic fuzzy systems have experimented with the provision of non-dominated trade-off solutions, each solution being a complete candidate rule set for the decision maker's consideration. These rule sets will range from the simplest and least accurate to the most complex and most accurate.

4.3 Part III: Adaptive Solution Schemes

These two chapters describe two different approaches to adaptive problem solving, involving mechanisms to select from a portfolio of algorithmic alternatives, adapting to the best choices for particular problems and instances.

Chapter 6: Exploring Hyper-Heuristic Methodologies with Genetic Programming

Hyper-heuristics represent a novel search methodology that is motivated by the goal of automating the process of selecting or combining simpler heuristics in order to solve hard computational search problems. This approach operates on a search space of heuristics rather than directly on a search space of solutions to the underlying problem which is the case with most meta-heuristics implementations. In this chapter, Edmund Burke, Mathew Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan and John Woodward look at the use of Genetic Programming to automatically generate heuristics for a given problem domain.

Chapter 7: Adaptive Constraint Satisfaction: The Quickest First Principle

James Borrett and Edward Tsang demonstrate the potential of adaptive constraint satisfaction in this chapter, using a technique known as algorithmic chaining. It is recognised that some constraint satisfaction instances are much easier to solve than others, and thus it makes sense to apply a simple and fast algorithm, whenever such an approach is adequate for solving the instance in question. However, when faced with exceptionally hard problem instances, a more complex (and slower) approach may be required. Algorithmic chaining presents a sequence of algorithms, which are applied to a problem instance in turn, if and when required. Thus, if the first algorithm is unsuccessful, the second in the sequence will be tried, and then the third, if required, and so on. The chapter describes the “Reduced Exceptional Behaviour Algorithm” (REBA), which is a technique based on algorithmic chaining. The REBA algorithm makes use of a mechanism for predicting when thrashing type behaviour is likely to occur, and results presented within the chapter clearly demonstrate the effectiveness of the approach in reducing susceptibility to exceptionally hard problem instances.

4.4 Part IV: Multi-Agent Systems

Multi-Agent Systems (MAS) provide increasingly popular paradigms for solving complex problems, using a distributed system of (simple) individual processing elements. These four chapters offer some novel solutions to difficult design and implementation issues associated with practical MAS.

Chapter 8: Collaborative Computational Intelligence in Economics

This chapter provides a general review of collaborative computational intelligence (CCI) in economics. Shu-Heng Chen demonstrates the potential of CCI by focussing on three research paradigms in economics: *heterogeneous agent-based economics*, *experimental economics*, and *financial data mining*. The essence of agent-based economics is a society of heterogeneous agents working together. Experimental economics is explored with respect to laboratories comprising both human agents and software agents. Finally, the chapter concludes with a survey of hybrid CI systems currently used in financial data mining.

Chapter 9: IMMUNE: A Collaborating Environment for Complex System Design

To address the dilemma of distributed versus central control in complex system design, decision support systems that enable robust collaboration amongst many design agents from different disciplines are required. The particular characteristics of such decision support systems must include immunity to catastrophic failures and sudden collapse that are usually observed in complex systems. This chapter, written by Mahmoud Efatmaneshnik and Carl Reidsema, lays the conceptual framework for IMMUNE as a robust collaborating design environment. Agents in IMMUNE are adaptive and can change their negotiation strategy and in this way can contribute to the overall capability of the design system to maintain its problem solving complexity.

Chapter 10: Bayesian learning for cooperation in multi-agent systems

Mair Allen-Williams and Nicholas R Jennings consider the problem of agent coordination in uncertain and partially observable systems. They present an approach to this problem using a Bayesian learning mechanism, and demonstrate its effectiveness on a cooperative scenario from the disaster response domain.

Chapter 11: Collaborative Agents for Complex Problems Solving

In a multi-agent system (MAS), agents that possess different expertise and resources collaborate together to handle problems which are too complex for individual agents. Generally, agent collaborations in a MAS can be classified into two groups, namely agent cooperation and agent competition. In this chapter Minjie Zhang, Quan Bai, Fenghui Ren and John Fulcher introduce two main approaches for complex problem solving via agent cooperation and/or competition, these being (i) a partner selection strategy among competitive agents, and (ii) dynamic team forming strategies among cooperative agents.

4.5 Part V: Computer Vision

Computer vision is a key application area for CI techniques. Chapters 12 and 13 discuss two extremely challenging applications: predicting human character traits from facial appearance and analyzing crowd dynamics, respectively.

Chapter 12: Predicting Trait Impressions of Faces Using Classifier Ensembles

Recent studies in social psychology indicate that people are predisposed to form impressions of a person's social status, abilities, dispositions, and character traits based on nothing more than that person's facial appearance. In this chapter Sheryl Brahnam and Loris Nanni present their work on building machine models of human perception, aimed at recognizing traits (such as dominance, intelligence, maturity, sociality, trustworthiness, and warmth) simply by observing human faces. They demonstrate that ensembles of classifiers work better than single classifiers, and also that ensembles composed of 100 Levenberg-Marquardt neural networks (LMNNs) seem to be as capable as most individual human beings are in their ability to predict the social impressions certain faces make on the average human observer.

Chapter 13: The Analysis of Crowd Dynamics: From Observations to Modelling

B. Zhan, P. Remagnino, D.N. Monekosso and S. Velastin describe how computer vision techniques, combined with statistical methods and a neural network, can be used to automatically observe, measure and learn crowd dynamics. New methods are proposed to measure crowd dynamics, and model the complex movements of a crowd.

4.6 Part VI: Communication for CI Systems

Distributed CI systems of all kinds need reliable, fast and efficient communications. These two chapters describe simple, low cost and effective ways to use the latest technology in a discriminatory way. Chapter 14 covers large scale collaborative sensor networks, and Chapter 15 focusses on opportunist networks.

Chapter 14 :Computational Intelligence for the Collaborative Identification of Distributed Systems

In this chapter Giorgio Biagetti, Paolo Crippa, Francesco Gianfelici and Claudio Turchetti suggest a new algorithm for the identification of distributed systems by large scale collaborative sensor networks. They describe how recent advances in hardware technologies have made it possible to realize low-power low-cost wireless devices and sensing units that are able to detect information from the distributed environment. Even though individual sensors can only perform simple local computation and communicate over a short range at low data rate, when deployed in large numbers they can form an intelligent collaborative network interacting with the surrounding environment in a large spatial domain. Sensor networks (SNs) characterized by low computational complexity, great learning capability, and efficient collaborative technology are highly desirable to discriminate, regulate and decide actions on real phenomena in many applications such as environmental monitoring, surveillance, factory instrumentation, defence and so on.

Chapter 15: Collaboration at the Basis of Sharing Focused Information: The Opportunistic Networks

This chapter is written by Bruno Apolloni, Guglielmo Apolloni, Simone Bassis, Gian Luca Galliani and Gianpaolo Rossi and discusses opportunistic networks. Opportunistic networks provide a communication protocol that is particularly suited to set up a robust collaboration within a very local community of agents. Like medieval monks who escaped world chaos and violence by taking refuge in small and protected communities, the authors point out that modern people may escape the information avalanche by forming virtual communities without relinquishing most of the benefits of the latest information and computer technology. A communication middleware to obtain this result is represented by opportunistic networks.

4.7 Part VII: Artificial Immune Systems

Chapter 16 provides a broad overview of artificial immune systems research, and focusses particularly on areas of natural immune systems that have been rather ignored by the AIS community in the past.

Chapter 16: Exploiting Collaborations in the Immune System: The Future of Artificial Immune Systems

This chapter, written by Emma Hart, Chris McEwan and Despina Davoudani, suggests some novel ways in which the natural immune system metaphor could be exploited to build new types of computational systems capable of meeting some of the challenges of the 21st Century, including self-configuration, self-maintenance, self-optimization and self-protection in an ever-changing environment. The authors focus particularly on aspects of the natural immune system which appear to have been largely overlooked by the artificial immune systems (AIS) research community in the past, and place significant emphasis on the design of *systems* rather than *algorithms*. The article puts forward some possible reasons why the potential promised by AIS has not yet been delivered, and suggests how this might be addressed in the future. The arguments are particularly relevant in light of recent advances in technology which present a new and challenging range of problems to be solved. A number of examples of systems in which steps are currently being taken to implement some of the mechanisms are then described. The chapter concludes with a discussion of an emerging field, that of *immuno-engineering* which promises a methodology which will facilitate maximum exploitation of immune mechanisms in the future.

4.8 Part VIII: Parallel Evolutionary Algorithms

Chapter 17 discusses the variety and importance of spatial interactions of populations in the natural world and demonstrates the relevance of these issues to parallel evolutionary algorithms.

Chapter 17: Evolutionary Computation: Centralized, Parallel or Collaborative

In this second chapter by Heinz Mühlenbein, the author focusses on the nature and importance of spatial interactions in evolutionary computation, and he also investigates cooperation and collaboration in this context. While “competition” is a fundamental component of Darwin’s theory of natural selection, it can be argued that cooperation and collaboration also play a large role in evolution and population dynamics. In this chapter genetic algorithms with several different spacial interaction schemes are tested, and the results are discussed in relation to Darwin’s ideas on the evolutionary gain achieved if subpopulations of individuals are periodically isolated from each other or from the main continental population of a species (i.e., the *continent-island cycle*).

4.9 Part IX: CI for Clustering and Classification

The four chapters in this section cover various aspects of pattern recognition, clustering and data mining.

Chapter 18: Fuzzy Clustering of Likelihood Curves for Finding Interesting Patterns in Expression Profiles

In this chapter Claudia Hundertmark, Lothar Jänsch and Frank Klawonn present a prototype-based fuzzy clustering approach that allows the automatic detection of regulatory regions within individual proteins. Cellular processes are mediated by proteins acting e.g. as enzymes (catalysts) in different metabolic pathways. Modifications are regularly made to specific regions of proteins within a living cell after that protein has been manufactured. The purpose of these post-translational modifications is to provide regulatory effects that will control the binding and activity properties of the modified proteins. In other words, the same protein will behave differently depending on the specific modifications made to it after its creation. Following the digestion of proteins into fragments (peptides), which is a necessary first stage of the work, the approach described in this chapter utilises likelihood curves to summarise the regulatory information of the peptides, based on a noise model obtained by an analytical process. Since the algorithm for the detection of peptide clusters is based on fuzzy clustering, their collaborative approach combines probabilistic concepts as well as principles from soft computing. However, fuzzy clustering is usually based on data points and its application to likelihood curves provided a considerable challenge for the authors. An interesting feature of this work is its potential transferability to noisy data from other applications, provided the noise can be specified by a noise model.

Chapter 19: A Hybrid Rule Induction/Likelihood Ratio-Based Approach for Predicting Protein-Protein Interactions

Mudassar Iqbal, Alex A. Freitas and Colin G. Johnson propose a new hybrid data mining method for predicting protein-protein interactions in this chapter. The purpose is to predict unknown protein interactions using relevant genomic information currently available. The new technique combines Likelihood-Ratios with rule induction algorithms and uses rule induction to discover the rules to partition the data. The discovered rules are subsequently interpreted as “bins” and used to compute likelihood ratios. In this way a rule induction algorithm learns classification rules, and these learned rules are used to improve the effectiveness of a likelihood ratio-based classifier, which is used to predict unknown protein interactions.

Chapter 20: Improvements in Flock-based Collaborative Clustering Algorithms

Esin Saka and Olfa Nasraoui begin their chapter with a brief survey of swarm intelligence clustering algorithms, and point out that since the early 90s, swarm intelligence (SI) has been a source of inspiration for clustering problems, and has been used in many applications ranging from image clustering to social clustering, and from document clustering to Web session clustering. The chapter then focuses mainly on a recent development: simultaneous data visualization and clustering using flocks of agents. The chapter presents some improvements to previous algorithms of this type and proposes a hybrid approach. Experiments on both artificial and real data confirm the validity of the approach and the advantages of the variants proposed in this chapter.

Chapter 21: Combining Statistics and Case-Based Reasoning for Medical Research

Case-based Reasoning (CBR) uses previous experience represented as cases to understand and solve new problems. A case-based reasoner remembers former cases similar to the current problem and attempts to modify solutions of former cases to fit the current problem. In this chapter Rainer Schmidt and Olga Vorobieva present a system, called ISOR, that helps to explain medical cases that do not fit a theoretical hypothesis. Indeed, it is often the case that no well-developed theory exists. Furthermore, at the start little knowledge or past experience may be available. This chapter focusses on the application of the ISOR system to the hypothesis that a specific exercise program improves the physical condition of dialysis patients. Additionally, for this application a method to restore missing data is presented.

Chapter 22: Collaborative and Experience-Consistent Schemes of System Modelling in Computational Intelligence

This study by Witold Pedrycz discusses a number of developments which form a conceptual and algorithmic framework for collaborative computational intelligence. First of all, the fundamentals of collaborative clustering are introduced in terms of information granules, i.e, fuzzy sets which emerge as a result of knowledge sharing. This is followed by the development of algorithmic definitions, which show the pertinent computing details. Hierarchies of clusters are also introduced, and experience-consistent fuzzy modeling is presented in the context of rule-based fuzzy models and neural networks.

References

1. Bezdek, J.C.: On the relationship between neural networks, pattern recognition and intelligence. International Journal of Approximate Reasoning 6, 85–107 (1992)
2. Bezdek, J.C.: What is computational intelligence? In: Zurada, J.M., Marks II, R.J., Robinson, C.J. (eds.) Computational Intelligence Imitating Life, pp. 1–12. IEEE Press, Los Alamitos (1994)
3. Deb, K.: Multi-objective optimization using evolutionary algorithms. John Wiley and Sons, Chichester (2001)
4. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: Optimization by a colony of cooperating agents. IEEE Trans. System Man Cybernetics Part B 26, 29–41 (1996)
5. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
6. Feigenbaum, E.A., McCorduck, P.: The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World. Addison-Wesley, Reading (1983)
7. Fogel, L., Owens, A., Walsh, M.: Artificial intelligence through simulated evolution. John Wiley, Chichester (1966)
8. Greenwood, G.W., Tyrrell, A.M.: Introduction to Evolvable Hardware: A Practical Guide for Designing Self-Adaptive Systems. Wiley-IEEE Press, Chichester (2006)
9. Holland, J.H.: Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor (1975)
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942–1948 (1995)
11. Koza, J.: Genetic programming. MIT Press, Cambridge (1992)
12. Langton, C. (ed.): Artificial life: An overview. MIT Press, Cambridge (1995)
13. Lenat, D.B.: Cyc: A Large-Scale Investment in Knowledge Infrastructure. Communications of the ACM 38(11) (November 1995)
14. Lighthill, J.: Artificial Intelligence: A General Survey. In: Artificial Intelligence: a paper symposium, Science Research Council, UK (1973)
15. McCarthy, J., Minsky, M.L., Rochester, N., Shannon, C.E.: A proposal for the Dartmouth summer research project on artificial intelligence, Stanford University, August 31 (1955), http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html
16. Mehrotra, K., Mohan, C.K., Ranka, S.: Elements of Artificial Neural Networks. MIT Press, Cambridge (1996)
17. Nguyen, T.H., Walker, E.A.: A First Course in Fuzzy Logic, 3rd edn. Chapman and Hall, Boca Raton (2006)
18. Pawlak, Z.: Rough sets. Int. J. Computer and Information Sci. 11, 341–356 (1982)
19. Rechenberg, I.: Cybernetic solution path of an experimental problem. Technical Report Translation number 1122, Ministry of Aviation, Royal aircraft Establishment, Farnborough, Hants, UK (1965)
20. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall, NJ (2003)
21. Winston, P.H.: Artificial Intelligence, 3rd edn. Addison Wesley, MS (1992)
22. Wooldridge, M.: An Introduction to MultiAgent Systems. John Wiley & Sons Ltd, NY (2002)
23. Zadeh, L.A.: Fuzzy Sets. Information and Control 8, 338–353 (1965)

Computational Intelligence: The Legacy of Alan Turing and John von Neumann

Heinz Mühlenbein

Abstract. In this chapter fundamental problems of collaborative computational intelligence are discussed. The problems are distilled from the seminal research of Alan Turing and John von Neumann. For Turing the creation of machines with human-like intelligence was only a question of programming time. In his research he identified the most relevant problems concerning evolutionary computation, learning, and structure of an artificial brain. Many problems are still unsolved, especially efficient higher learning methods which Turing called initiative. Von Neumann was more cautious. He doubted that human-like intelligent behavior could be described unambiguously in finite time and finite space. Von Neumann focused on self-reproducing automata to create more complex systems out of simpler ones. An early proposal from John Holland is analyzed. It centers on adaptability and population of programs. The early research of Newell, Shaw, and Simon is discussed. They use the logical calculus to discover proofs in logic. Only a few recent research projects have the broad perspectives and the ambitious goals of Turing and von Neumann. As examples the projects Cyc, Cog, and JANUS are discussed.

1 Introduction

Human intelligence can be divided into *individual*, *collaborative*, and *collective intelligence*. Individual intelligence is always multi-modal, using many sources of information. It developed from the interaction of the humans with their environment. Based on individual intelligence, collaborative intelligence developed. This means that humans work together with all the available allies to solve problems. On the next level appears collective intelligence. It describes the phenomenon that families, groups, organizations and even entire societies seem to act as a whole living organism.

Heinz Mühlenbein
Fraunhofer Institut Autonomous intelligent Systems Schloss Birlinghoven 53757
Sankt Augustin, Germany
e-mail: heinz.muehlenbein@online.de

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 23–43.
springerlink.com

The importance of interactions between higher animals has been reinforced by the discovery of *mirror neurons*. These are neurons which fire both when an animal acts and when the animal observes the same action performed by another (especially conspecific) animal. The neurons have been observed in primates and are believed to exist also in humans and in some birds. The function of the mirror system is still a subject of much speculation. To date no plausible neural or computational models have been developed to explain how mirror neurons support the cognitive functions.

In my opinion the most impressive collaborative computational intelligence examples developed so far are the search machine Google and Wikipedia. In both systems the interaction human-computer plays an important role. Google is a gigantic storage system with an impressive fast search engine. It remains the task of the user to filter out the important information from the search results.

Wikipedia tries to make the dream of the Age of Enlightenment become true, to develop an encyclopedia describing all human knowledge and making it accessible to all humans. Both systems use pure text driven search. More intelligent search methods have been not successful so far. Despite the many efforts no computational system is approaching the level of human intelligence.

Today computational intelligence is partitioned into many specialized and separate research areas. This was not always the case. The aim of this chapter is to recall the broader issues and research goals of computational intelligence. To this end the seminal research of Alan Turing and John von Neumann is reviewed in detail. Their proposals discuss many areas of computational intelligence necessary to create automata with human-like intelligence.

Right at the beginning of electronic computers researchers looked into nature for ideas to solve difficult problems or even create what is called today *artificial intelligence*. Because of the lack of understanding the functioning of natural systems, the research had to be largely experimental. This was already pointed out by John von Neumann [25].

> *Natural organism are, as a rule, much more complicated and subtle, and therefore much less well understood in detail, than are artificial automata. Nevertheless, some regularities, which we observe in the organization of the former may be quite instructive in our thinking and planning of the latter; and conversely, a good deal of our experiences and difficulties with our artificial automata can be to some extend projected on our interpretations of natural organisms.*

In this chapter I will first review the work of Alan Turing, described in his famous seminal paper "Computing Machinery and Intelligence" [23] and in the not so well known paper "Intelligent Machinery" [24]. Turing's thoughts about *learning, evolution, and structure of the brain* are described.

Then I will discuss the most important paper of John von Neumann concerning our subject "The General and Logical Theory of Automata" [25]. John von Neumann's research centers on *artificial automata, computability, complexity, and self-reproduction*

All three papers were written before the first electronic computers became available. Turing even wrote programs for paper machines. As a third example I will describe the proposal of John Holland [10]. The simplification of this proposal lead later to the famous genetic algorithm [11]. The historical part ends with a discussion of the early research of Newell, Shaw and Simon.

I will first discuss this early research in detail, without reference to today's knowledge. Then I will try to evaluate the proposals in answering the following questions

- What are the major ideas for creating machine intelligence?
- Did the original proposals lack important components we see as necessary today?
- What are the major research problems of the proposals and do solutions exist today?

Then two recent large projects are shortly summarized. The goal of the project Cyc is to specify *common sense knowledge in a well-designed language* The Cog project tried to build a *humanoid robot that acts like a human*. In addition the architecture of our *hand-eye robot JANUS* is described. It has a modular structure similar to the human brain.

This chapter is a tour de force in computational intelligence. It requires that the reader is willing to contemplate fundamental problems arising in building intelligent machines. Solutions are not given. I hope that the reader finds interesting research problems worthy of being investigated. This paper extends my research started in [15].

2 Turing and Machine Intelligence

The first sentences of the paper ”Computing machinery and intelligence” have become famous.

I propose to consider the question “Can machines think?” This should begin with definitions of the meaning of the terms “machine” and “think”....But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words. The new form of the question can be described in terms of a game which we call the imitation game.

Turing's definition of the imitation game is more complicated than that normally used today. Therefore I will describe it shortly. It is played with three actors, a man (A), a woman (B) and an interrogator (C). The object of the game for the interrogator is to determine which of the other two is the man and which is the woman. It is A's objective in the game to cause C to make the wrong identification. Turing then continues: “We now ask the question ‘What will happen when a machine takes the part of A in the game?’” Will the interrogator decide wrongly as often when the game is played like this as he does when the game is played between a man and a woman? These questions will replace our original “Can machines think”.

Why did Turing not define just a game between a human and a machine trying to imitate a human, as the Turing test is described today? Is there an additional trick in introducing gender into the game? There has been quite a lot of discussions as to whether this game characterizes human intelligence at all. Its purely behavioristic definition leaves out any attempt to identify important components which together produce human intelligence. I will not enter this discussion here, but just state the opinion of Turing about the outcome of the imitation game.

I believe that in about fifty years' time it will be possible to programme computers with a storage capacity of about 10^9 *bits to make them play the imitation game so well that an average interrogator will not have more than* 70% *chance of making the right identification after five minutes of questioning.*

The very detailed prediction is funny: Why a 70% chance, why a duration of five minutes? In the next section I will discuss what arguments Turing used to support this prediction.

2.1 Turing's Construction of an Intelligent Machine

In Sections 2 – 6 of [23] Turing mainly seeks to refute general philosophical arguments against the possibility of constructing intelligent machines. "The reader will have anticipated that I have no very convincing argument of a positive nature to support my views. If I had I should not have taken such pains to point out the fallacies in contrary views. Such evidence as I have I shall now give." What is Turing's evidence?

As I have explained, the problem is mainly one of programming. Advances in engineering will have to be made too, but it seems unlikely that these will not be adequate for the requirements. Estimates of the storage capacity of the brain vary from 10^{10} *to* 10^{15} *binary digits.*[1] *I incline to the lower values and believe that only a small fraction is used for the higher types of thinking. Most of it is probably used for the retention of visual impressions. I should be surprised if more than* 10^9 *was required for satisfactory playing of the imitation game. Our problem then is to find out how to programme these machines to play the game. At my present rate of working I produce about a thousand digits of programme a day, so that about sixty workers, working steadily through fifty years might accomplish the job, if nothing went into the wastepaper basket.*

The time to construct a machine which passes the imitation game is derived from an estimate of the storage capacity of the brain[2] and the speed of programming.

[1] At this time the number of neurons was estimated as being between 10^{10} to 10^{15}. This agrees with the estimates using today's knowledge.

[2] It was of course a mistake to set the storage capacity equal to the number of neurons! Von Neumann estimated the storage capacity of the brain to be about 10^{20}. But this affects in no way the logic of Turing's argument.

Turing did not see any problems in creating machine intelligence purely by programming, he just found it too time consuming. So he investigated if there exist more expeditious methods. He observed:

"In the process of trying to imitate an adult human mind we are bound to think a good deal about the process which has brought it to the state that it is in. We may notice three components.

1. The initial state of the brain, say at birth.
2. The education to which it has been subjected.
3. Other experience, not to be described as education, to which it has been been subjected.

Instead of trying to produce a programme to simulate an adult mind, why not rather try to produce one which simulates the child's...Presumably the child brain is something like a notebook. Rather little mechanism, and lots of blank sheets. Our hope is that there is so little mechanism in the child brain that something like it can easily be programmed. The amount of work in the education we can assume, as a first approximation, to be much the same as for the human child."

2.2 Turing on Learning and Evolution

In order to achieve a greater efficiency in constructing a machine with human like intelligence, Turing divided the problem into two parts

- The construction of a child brain
- The development of effective learning methods

Turing notes that the two parts remain very closely related. He proposes to use experiments: teaching a child machine and seeing how well it learns. One should then try another and see if it learns better or worse. "There is an obvious connection between this process and evolution, by the identifications

- structure of the machine = hereditary material
- changes of the machine = mutations
- Natural selection = judgment of the experimenter

Survival of the fittest is a slow process of measuring advantages. The experimenter, by the exercise of intelligence, should be able to speed it up."

Turing then discusses learning methods. He notes ([23], p.454): "We normally associate the use of punishments and rewards with the teaching process...The machine has to be so constructed that events which shortly proceeded the occurrence of a punishment signal are unlikely to be repeated, whereas a reward signal increases the probability of repetition of the events which lead to it."

But Turing observes the major drawback of this method: "The use of punishments and rewards can at best be part of the teaching process. Roughly speaking, if the teacher has no other means of communicating to the people, the amount of information which can reach him does not exceed the total number of rewards and punishments applied."

In order to speed up learning Turing demanded that the child machine should understand some language. In the final pages of the paper Turing discusses the problem of the complexity the child machine should have. He proposes to try two alternatives: either to make it as simple as possible to allow learning or to include a complete system of logical inference. He ends his paper with the remarks: “Again I do not know the answer, but I think both approaches should be tried. We can see only a short distance ahead, but we can see plenty there that needs to be done.”

2.3 Turing and Neural Networks

In the posthumously published paper *Intelligent Machinery* [24] Turing describes additional details how to create an intelligent machine. First he discusses possible components of a child machine. He introduces *unorganized machines* of type A, B, and P. A and B are artificial neural networks with random connections. They are made up from a rather large number N of similar units, which can be seen as binary neurons. Each unit has two input terminals and one output terminal which can be connected to the input terminals of 0 (or more) other units. The connections are chosen at random. All units are connected to a central synchronizing unit from which synchronizing pulses are emitted. Each unit has two states. The dynamics is defined by the following rule:

The states from the units from which the input comes are taken from the previous moment, multiplied together and the result is subtracted from 1.

Thus a neuron is nothing else than a NAND gate. The state of the network is defined by the states of the units. Note that the network might have lots of loops, it continually goes through a number of states until a period begins. The period cannot exceed 2^N cycles. In order to allow learning the machine is connected with some input device which can alter its behavior. This might be a dramatic change of the structure, or changing the state of the network.

Maybe Turing had the intuitive feeling that the basic transition of the type A machine is not enough, therefore he introduced the more complex B-type machine. I will not describe this machine here, because neither for the A or the B machine did Turing define precisely how learning can be done.

A learning mechanism is introduced with the third machine, called a P-type machine. The machine is an automaton with a number of N configurations. There exists a table where, for each configuration, the action the machine has to take is specified. The action may be either

1. To do some externally visible act $A_1, \ldots A_k$
2. To set a memory unit M_i

The reader should have noticed that the next configuration is not yet specified. Turing surprisingly defines: If the current configuration is s, then the next configuration is the remainder of $2s$ or $2s+1$ on division by N. These two configurations are called the alternatives 0 and 1. The reason for this definition is the learning mechanism Turing defines. At the start the description of the machine is largely incomplete. The entries for each configuration might be in five states, either U (uncertain), or T0 (try alternative 0), T1 (try alternative 1), D0 (definite 0) or D1 (definite 1).

Learning changes the entries as follows: If the entry is U, the alternative is chosen at random, and the entry is changed to either T0 or T1 according to whether 0 or 1 was chosen. For the other four states, the corresponding alternatives are chosen. When a pleasure stimulus occurs, state T is changed to state D, when a pain stimulus occurs, T is changed to U. Note that state D cannot be changed. The proposed learning method sounds very simple, but Turing surprisingly remarked:

I have succeeded in organizing such a (paper) machine into a universal machine.

Today the universal machine is called the *Turing Machine*. Turing even gave some details of this particular P-type machine. Each instruction consisted of 128 digits, forming four sets of 32 digits, each of which describes one place in the main memory.

2.4 Discipline and Initiative

We now turn to the next important observation of Turing. Turing notes that punishment and reward are very slow learning techniques. So he requires:

If the untrained infant's mind is to become an intelligent one, it must acquire both discipline and initiative.

Discipline means strictly obeying the punishment and reward. But what is initiative? The definition of initiative is typical of Turing's behavioristic attitude. “Discipline is certainly not enough in itself to produce intelligence. That which is required in addition we call initiative. This statement will have to serve as a definition. Our task is to discover the nature of this residue as it occurs in man, and to try and copy it in machines.”

With only a paper computer available Turing was not able to investigate the subject initiative further. Nevertheless he made the bold statement [24]: “A great positive reason for believing in the possibility of making thinking machinery is the *fact that it is possible to make machinery to imitate any small part of a man*. One way of setting about our task of building a thinking machine would be to take a man as a whole and to try to replace all parts of him by machinery...Thus although this method is probably the ‘sure’ way of producing a thinking machine it seems to be altogether too slow and impracticable. Instead we propose to try and see what can be done with a ‘brain’ which is more or less without a body providing, at most organs of sight, speech, and hearing. We are then faced with the problem of finding suitable branches of thought for the machine to exercise its powers in.”

Turing mentions the following fields as promising:

- Various games, e.g. chess, bridge
- The learning of languages
- Translation of languages
- Cryptography
- Mathematics

Turing remarks: "The learning of languages would be the most impressive, since it is the most human of these activities. This field seems however to depend rather too much on sense organs and locomotion to be feasible." Turing seems here to have forgotten that language learning is necessary for his imitation game!

3 Von Neumann's Logical Theory of Automata

In 1938 Alan Turing was assistant to John von Neumann. But later they worked completely independently from each other, not knowing the thoughts the other had concerning the possible applications of the newly designed electronic computers. A condensed summary of the research of John von Neumann concerning machine intelligence is contained in his paper "The General and Logical Theory of Automata" [25]. This paper was presented in 1948 at the Hixon symposium on: *Cerebral mechanism of behavior*. Von Neumann was the only computer scientist at this symposium. The reason was that von Neumann closely observed the theoretical research aimed to understand the brain in order to use the results for artificial automata.

Von Neumann notices three major limitations of the present size of artificial automata

- The size of componentry
- The limited reliability
- The lack of a logical theory of automata

There have been tremendous achievements in the first two areas. Therefore I will concentrate on the theory problem. Here von Neumann predicted:

The logic of automata will differ from the present system of formal logic in two relevant respects.

1. *The actual length of "chains of reasoning", that is, of the chains of operations, will have to be considered.*
2. *The operations of logic will all have to be treated by procedures which allow exceptions with low but non-zero probabilities.*

...This new system of formal logic will move closer to another discipline which has been little linked in the past with logic. This is thermodynamics, primarily in the form it was received from Boltzmann, and is that part of theoretical physics which comes nearest in some of its aspects to manipulating and measuring information.

Von Neumann tried later to formalize probabilistic logic. His results appeared in [26]. But this research was more or less a dead end, because von Neumann did not abstract from the hardware components. They are unreliable and have a certain probability of failure. In addition, von Neumann included time in his model, making a mathematical analysis of a given system difficult. Probabilistic reasoning is now heavily used in artificial intelligence [17]. The chains of operations are investigated in a branch of theoretical computer science called computational complexity [8].

3.1 McCulloch-Pitts Theory of Formal Neural Networks

In 1943 McCulloch and Pitts [13] had described the brain by a formal neural network, consisting of interconnected binary neurons. Von Neumann summarizes their major result follows:

"The 'functioning' of such a network may be defined by singling out some of the inputs of the entire system and some of its outputs, and then describing what original stimuli on the former are to cause what ultimate stimuli of the latter. McCulloch and Pitts' important result is that any functioning in this sense which can be defined at all logical, strictly, and unambiguously in a finite number of words can also be realized by such a formal system."

In modern terms: Any computable function can be realized by a sufficiently large McCulloch and Pitts network.
McCulloch and Pitts had derived this result by showing that their formal neural network connected to an infinite tape is equivalent to a Turing machine. But even given this result, von Neumann observes that at least two problems remain

1. Can the network be realized within a practical size?
2. Can every existing mode of behavior really be put completely and unambiguously into words?

Von Neumann informally discusses the second problem, using the problem of analogy. He remarks prophetically:

There is no doubt that any special phase of any conceivable form of behavior can be described "completely and unambiguously" in words.... It is, however an important limitation, that this applies only to every element separately, and it is far from clear how it will apply to the entire syndrome of behavior.

This severe problem has not been noticed by Turing. Using the example visual analogy von Neumann argues: "One can start describing to identify any two rectilinear triangles. These could be extended to triangles which are curved, whose sides are only partially drawn etc... We may have a vague and uncomfortable feeling that a complete catalogue along such lines would not only be exceedingly long, but also unavoidably indefinite at its boundaries. All of this, however, constitutes only a small fragment of the more general concept of identification of analogous geometrical objects. This, in turn, is only a microscopic piece of the general concept of visual analogy." Thus von Neumann comes to the conclusion:

Now it is perfectly possible that the simplest and only practical way to say what constitutes a visual analogy consists in giving a description of the connections of the visual brain....It is not at all certain that in this domain a real object might not constitute the simplest description of itself.

Von Neumann ends the section with the sentence: “The foregoing analysis shows that one of the relevant things we can do at this moment is to point out the directions in which the real problem does not lie.” In order to understand and investigate the fundamental problem, von Neumann identified an important subproblem. In nature it is obvious that more complex beings have been developed from less complex ones. Is this also possible using automata? How much complexity is needed for automata to create more complex ones?

3.2 Complication and Self-reproduction

Von Neumann starts the discussion of complexity with the observation that if an automaton has the ability to construct another one, there must be a decrease in complication. In contrast, natural organisms reproduce themselves, that is, they produce new organisms with no decrease in complexity. So von Neumann tries to construct a general artificial automata which could reproduce itself. The famous construction consists of the following automata:

1. A general constructive machine, A, which can read a description $\Phi(X)$ of another machine, X, and build a copy of X from this description:

$$A + \Phi(X) \rightsquigarrow X$$

2. A general copying machine, B. which can copy the instruction tape:

$$B + \Phi(X) \rightsquigarrow \Phi(X)$$

3. A control machine, C, which when combined with A and B, will first activate B, then A, link X to $\Phi(X)$ and cut them loose from A+B+C

$$A + B + C + \Phi(X) \rightsquigarrow X + \Phi(X)$$

Now choose X to be A+B+C

$$A + B + C + \Phi(A + B + C) \rightsquigarrow A + B + C + \Phi(A + B + C)$$

4. It is possible to add the description of any automaton D

$$\begin{aligned} A + B + C + \Phi(A + B + C + D) \rightsquigarrow A + B + C + D \\ + \Phi(A + B + C + D) \end{aligned}$$

Now allow mutation on the description $\Phi(A + B + C + D)$

$$A+B+C+\Phi(A+B+C+D') \rightsquigarrow A+B+C+D' \\ +\Phi(A+B+C+D')$$

Mutation at the D description will lead to a different self-reproducing automaton. This might allow the system to simulate some kind of evolution as seen in natural organisms.

Von Neumann later constructed a self-reproducing automata which consisted of 29 states [27]. This convinced von Neumann that complication can also be found in artificial automata. Von Neumann ends the paper with the remark:

This fact, that complication, as well as organization, below a critical level is degenerative, and beyond that level can become self-supporting and even increasing, will clearly play an important role in any future theory of the subject.

Von Neumann was well aware of the other two important evolutionary processes besides replication - namely variation and selection. He decided that knowledge about these two processes was not yet sufficient to incorporate them in his theory of automaton. “Conflicts between independent organisms lead to consequences which, according to the theory of natural selection, are believed to furnish an important mechanism of evolution. Our models lead to such conflict situations. The conditions under which this motive for evolution can be effective here may be quite complicated ones, but they deserve study.”

Cellular automata have lead to great theoretical research. They can easily be extended to have the power of Turing machines. Nevertheless, the central problem of this approach remains unsolved: How can the automata evolve complex problem solving programs starting with fairly simple initial programs? This happened in biological evolution. Starting with small self-reproducing units complex problem solving capabilities have evolved, culminating in the human brain.

4 Holland’s Logical Theory of Adaptive Systems

In the paper “Outline for a Logical Theory of Adaptive Systems” [10] John Holland tried to continue the scientific endeavor initiated by von Neumann. He wrote:

The theory should enable to formulate key hypotheses and problems particularly from molecular control and neurophysiology. The work in theoretical genetics should find a natural place in the theory. At the same time, rigorous methods of automata theory, particularly those parts concerned with growing automata should be used.

Holland’s proposal is a very early attempt to work on a constructive theory of the evolution of automata. It tries to combine *being, acting, developing, and evolving* (see Chapter 17 for more details). This proposal is so important that I will describe it in detail. Holland’s emphasis (like von Neumann‘s) is foremost on theories and systems, he does not claim to solve grand challenge applications with the proposed methods. This can be tried after the theories have been formulated and verified.

"Unrestricted adaptability (assuming nothing is known of the environment) requires that the adaptive system be able initially to generate any of the programs of some universal computer ... With each generation procedure we associate the population of programs it generates;... In the same vein we can treat the *environment as a population of problems*."

Now let us have a closer look at Holland's model. First, there is a finite set of generators (programs) $(g_1,\ldots,g_k)$. The generation procedure is defined in terms of this set and a *graph* called a *generation tree.* Each permissible combination of generators is represented by a vertex in the generation tree. Holland now distinguishes between auxiliary vertices and main vertices. Each auxiliary vertex will be labeled with two numbers, called the *connection* and *disconnection probabilities.* This technique enables to create new connections or to delete existing connections. Each main vertex is labeled with a variable referred to as *density*. The interested reader is urged to read the original paper [10].

Holland claims that from the generation tree and the transition equations of any particular generation procedure, one can *calculate the expected values of the densities of the main vertices as a function of time.* Holland writes: "From the general form of the transition equations one can determine such things as conditions under which the resulting generation procedures are *stationary processes.*" Thus Holland already tried to formulate a stochastic theory of program generation! This is an idea still waiting to be explored.

Holland's next extension of the system is similar in spirit to von Neumann's self-reproducing automata. Holland introduces *supervisory programs* which can construct *templates* which alter the probabilities of connections. Templates play the role of catalysts or enzymes. Thus program construction is also influenced by some kind of "chemical reactions."

The above process is not yet adaptive. Adaptation needs an environment posing problems. Therefore Holland proposes that the *environment is treated as a population of problems*. These problems are presented by means of a finite set of initial statements and an algorithm for checking whether a purported solution of the problem is in fact a solution. Holland then observes the problems of partial solutions and subgoals. "When we consider the interaction of an adaptive system with its environment we come very soon to questions of partial solutions, subgoals etc. *The simplest cases occur when there is an a priori estimate of the nature of the partial solution and a measure of the closeness of its approach to the final solution*."

Holland then observes that a *rich environment* is crucial for the adaptation. "Mathematical characterization of classes of rich environments relative to a given class of adaptive systems constitutes one of the major questions in the study of adaptive systems. ... An adaptive system could enhance its rate of adaptation by somehow enriching the environment. Such enrichment occurs if the adaptive system can generate subproblems or subgoals whose solution will contribute to the solution of the given problems of the environment."

It is very interesting to note that Holland distinguished three kinds of programs – supervisory programs, templates, and programs for the problem solution. The supervisory programs use a probabilistic generation tree to generate programs, the templates are used as catalyst to "skew" the generation process. Holland perceived a hierarchy of programs [9]:

1. productive systems – the generator system is able to produce other generators
2. autocatalytic systems – the generator system produces generators which are used in the construction
3. self-duplicating systems – the generator system produces duplicates of itself
4. general adaptive systems – has still to be defined

"The beginning of such a definition (of adaptive systems) lies in the following consideration: with the help of concepts such as autocatalytic and self-duplicating generator systems it is possible to define such concepts as steady-state equilibria and homeostasis for embedded automata...If the generator system for such an automaton has a hierarchical structure, then a small change in structure produces a small change in proportion to the "position" of the change in the hierarchy...By making changes first at the highest level and then at progressively lower levels of the hierarchy, it should be possible to narrow down rather quickly to any automaton in this category having some initially prescribed behavior."

I believe that Holland's proposal is a very good starting point for future research. It puts forward many ideas not yet contained in current research. After working for several years on this theory Holland turned to a much simpler evolution model. The environment is hidden in a *fitness function*. Evolution then reduces to an optimization problem. This research lead to the famous *genetic algorithm*.

5 The Beginning of Artificial Intelligence - The Logic Theorist

The term artificial intelligence was coined in the mid fifties. One of the first achievements was the logic theory machine, also called the Logic Theorist LT by Newell, Shaw and Simon [16]. LT proved theorems in elementary symbolic logic, more precisely the sentential calculus. It consists of expressions built from combinations of basic symbols. *Principia Mathematica* from Russell and Whitehead lists five expressions as axioms for the sentential calculus. The first three are

$$\begin{aligned}(p \; or \; q) &\rightarrow p \\ p &\rightarrow (p \; or \; q) \\ (p \; or \; q) &\rightarrow (q \; or \; p)\end{aligned}$$

p and q are binary variables. Given any variable p we can form $(not\ p)$ Given any two variables we can form the expression $(p\ or\ q)$ or $p \rightarrow q$. From these axioms theorems can be derived.

When the LT found a simpler proof of proposition 2.85 of Principia Mathematica, Simon wrote to Russell: "We have accumulated some interesting experience about the effects of simple learning programs superimposed on the basic performance program. For example we obtain rather striking improvements in problem-solving ability by inducing the machine to remember and use the fact that particular theorems have proved in the past useful to it in the connection with particular proof methods.....In general, the machine's problem solving is much more elegant when it works with a selected list of strategic theorems than when it tries to remember and use all the previous theorems" ([20] ,p.208).

Russell answered: "I am delighted by your example of superiority of your machine to Whitehead and me...I am also delighted by your exact demonstration of the old saw that wisdom is not the same thing as erudition" ([20], p. 208).

Simon made serious attempts to interpret LT as a psychological theory of problem solving. But after analyzing thinking-aloud protocols he realized that LT did not yet fit at all the detail of human problem-solving revealed by the protocols. Newell and Simon identified the subjects principal problem solving tool. They called it *means-ends analysis*.

Means-ends analysis is accomplished by comparing the problem goal with the present situation and noticing one or more differences between them. The observed difference jogs memory for an action that might reduce or eliminate the differences. The action is taken, a new situation is observed, and if the goal has still not been reached, the whole process is repeated. Means-ends analysis is used today in many problem solving tools. In principle backpropagation in artificial neural networks can also be seen as means-ends analysis.

Means-ends analysis is the central component of the next AI system Newell, Shaw, and Simon developed. It was named the *General Problem Solver* GPS. It is an impressive system incorporating many important problem solving techniques, but difficult applications have not been reported.

The success of LT lead Simon and Newell in 1958 to their famous prediction : "I do not want to shock you, but there are now in the world machines that think, that learn, and that create. Moreover, their ability to do these things is going to increase rapidly until - in a visible future - the range of problems they can handle will be coextensive with the range to which the human mind has been applied [19]."

6 Discussion of the Early Proposals to Create Artificial Intelligence by Simulating Evolution

I have reviewed only four of the early proposals which simulate natural systems to create machine intelligence. One observation strikes immediately: all the researchers investigated the problem of machine intelligence on a very broad scale. The main emphasis of Turing was the design of efficient learning schemes. For Turing it was obvious that only by efficient learning of something like a child machine an intelligent machine could be developed. The attitude of Turing was purely that of a computer scientist. He firmly believed that machine intelligence equal to or surpassing human intelligence could eventually be created.

Von Neumann's approach was more interdisciplinary, using also results from the analysis of the brain. He had a similar goal, but he was much more cautious concerning the possibility of creating an automaton with intelligence. He identified important problems which blocked the road to machine intelligence.

Both von Neumann and Turing investigated formal neural networks as a basic component of an artificial brain. This component was not necessary for the design, it was used only to show that the artificial automata could have a similar organization as the human brain. Both researchers ruled out that a universal theory of intelligence could be found, which would make it possible to program a computer according to this theory. So Turing proposed to use *learning* as the basic mechanism, von Neumann *self-reproducing automata*.

Von Neumann was sceptical about the creation of machine intelligence. He was convinced that learning leads to the *curse of infinite enumeration*. While every single behavior can be unambiguously described, there is obviously an infinite number of different behaviors. Turing also saw the limitations of teacher based learning by reward and punishment, therefore he required that the machine needs *initiative* in addition. Turing had no idea how learning techniques for initiative could be implemented. He correctly observed that it was necessary for creating machine intelligence by learning. Higher-level learning methods are still an open research problem.

The designs of Turing and von Neumann contain all components considered necessary today for creating machine intelligence. Turing ended his investigation with the problem of learning by initiative. Von Neumann invented as a first step self-reproducing cellular automata.

There is no major flaw in their designs. Von Neumann's question - can visual analogy be described in finite time and limited space, is still unsolved.

In order to make the above problem clear, let me formulate a conjecture: The computational universe can be divided into three sectors: *computable problems*; *non-computable problems* (that can be given a finite, exact description but have no effective procedure to deliver a definite result); and, finally, problems whose individual behaviors are, in principle, computable, but that, in practice, we are *unable to formulate in an unambiguous language* understandable for a Turing machine. Many non-computable problems are successfully approached by heuristics, but it seems very likely that the problem of visual analogy belongs to the third class.

Holland proposed a general scheme for breeding intelligent programs using the mechanisms of evolution. This was the most ambitious proposal using program generation by evolutionary principles to create intelligent machines. This proposal tried to circumvent Turing's problem to code all the necessary knowledge.

Let us try to contrast the approach of Turing with those of von Neumann and Holland. Turing proposed to programme the knowledge the humans have. In order to speed up the implementation he suggested to programme an automaton with only child like intelligence. The automaton child is then taught to become more intelligent.

Von Neumann was skeptical if all the components necessary for human like intelligence could be programmed in finite time and finite space. Therefore von Neumann started with the idea to automatically evolve automata. This idea was extended by Holland proposing an environment of problems to evolve the automata. On first sight this seems to solve the programming problem. Instead of copying human like intelligence, an environment of problems was used. But Holland overlooked the *complexity of programming the problems*. This would seem to be no easier than programming the knowledge humans have about the environment.

Holland's proposal to use stochastic systems, their steady-state equilibria and homeostasis is in my opinion still a very promising approach for a constructive evolution theory of automata. Holland himself never implemented his general model. It is still a theoretical design.

Later von Neumann's proposal has been extended insofar as both, the *problem solving programs and the problems evolve together* [14]. This obviously happened in natural evolution. In a new research discipline called *artificial life* several attempts have been made to evolve automata and the environment together, but the evolution always stopped very early.

Newell, Shaw and Simon concentrated on the higher level problem solving capabilities of humans. Evolutionary principles or lower level structures like the human brain are not considered to be relevant. Instead a theory of problem solving by humans is used. Their research lead to cognitive science and to artificial intelligence research based on theories of intelligence. Despite their great optimism, no convincing artificial intelligence system has been created so far using this approach.

7 Cyc and Cog: Two Large Projects in the Legacy of Alan Turing

Only very few big projects have been pursued in the spirit of Alan Turing. Two recent examples are the projects Cyc and Cog. Cyc is an attempt to assemble a comprehensive ontology and data base of everyday knowledge, with the goal of enabling the system human-like reasoning. The goal of the Cog project was to create a humanoid robot.

7.1 The Cyc Project

The Cyc project was started in 1984 with the goal to specify *common sense knowledge in a well designed language* [12, 6]. Cyc attempts to assemble a comprehensive ontology and database of everyday common sense knowledge, with the goal of enabling AI applications to perform human-like reasoning. The original knowledge base is proprietary, but a smaller version of the knowledge base, intended to establish a common vocabulary for automatic reasoning, was released 2005 as OpenCyc under an open source license.

Typical pieces of knowledge represented in the database are "Every tree is a plant" and "Plants die eventually". When asked whether trees die, the inference engine can draw the obvious conclusion and answer the question correctly. The Knowledge Base (KB) contains over a million human-defined assertions, rules or common sense ideas. These are formulated in the language CycL, which is based on predicate calculus and has a syntax similar to that of the Lisp programming language.

Much of the current work on the Cyc project continues to be knowledge engineering, representing facts about the world by hand, and implementing efficient inference mechanisms on that knowledge. Increasingly, however, work at Cycorp involves giving the Cyc system the ability to communicate with end users in natural language, and to assist with the knowledge formation process via machine learning.

Currently (2007) the knowledge base consists of

- 3.2 million assertions (facts and rules)
- 280,000 concepts
- 12,000 concept-interrelating predicates

Cyc runs now for 32 years, it is the longest running project in the history of AI. But despite its huge effort its success is still uncertain. Up to now Cyc has not been successfully used for any broad AI application. The system is far away from being used for a Turing test.

We remind the reader, that the coding of knowledge was considered by Turing as too inefficient. Von Neumann even doubted if the necessary knowledge for visual analogy could be specified in finite time. Today Cyc seems to be more a confirmation of von Neumann's doubt than a refutation.

7.2 The Cog Project

The Cog project was started in 1993 with extreme publicity. The goal was to understand human cognitive abilities well enough to build a humanoid robot that develops and acts similar to a person [3, 4]. One of the key ideas of the project was to build a robot with capabilities similar to a human infant. We have encountered this idea already in Turing's proposal.

"By exploiting a gradual increase in complexity both internal and external, while reusing structures and information gained from previously learned behaviors, we hope to be able to learn increasingly sophisticated behavior [4]." Cog was designed bottom-up [3]. This lead to reasonable success in the beginning. The big problems appeared later.

Brooks et al. wrote prophetically: *To date (1999), the major missing piece of our endeavor is demonstrating coherent global behavior from existing subsystems and sub-behaviors. If all of these systems were active at once, competition for actuators and unintended couplings through the world would result in incoherence and interference among the subsystems [4].*

During the course of the project a lot of interesting research has been done. But the problem of coherent or even intelligent behavior could not be solved. Therefore the project was stopped in 2002 without even entering the learning or development phase.

8 The JANUS Hand-Eye Robot and the Pandemonium Architecture

With my research group I have also tried two larger research projects in the spirit of Alan Turing and John von Neumann. The most spectacular was our hand-eye robot JANUS. With JANUS we bridged the gap between small-scale neural networks and real-world applicability. The robot had two eyes and two arms with which it observed and manipulated its environment. The robot learned from experience and self-supervision, initialized only with a few essential properties. JANUS also incorporated structural and functional medical knowledge of the brain.

The JANUS architecture was directly influenced by the top-level structure of the human brain and its hemispherical functional lateralization [7, 22]. However the similarities end at that level and a great deal of freedom is permitted in lower-level neural networks. The name JANUS was chosen after the Roman god for a specific reason: The brain not only looks out and observes and weighs up its environment, but it also looks *inwardly* and is aware of its own processes. It has a *reflective architecture*.

The JANUS brain controls a physical robot that may exist in a changing environment. The robot can be affected by the environment either directly, through physical contact with objects, or indirectly by the thought and learning processes. The highest level description of the JANUS architecture is illustrated in Figure 1.

The brain is divided in two halves laterally and two blocks vertically. All sensory signals from the left side of the robot pass directly to the right half of the brain, while those from the right side pass directly to the left half of the brain. The left half of the brain controls the motor outputs affecting the right side of the robot, and similarly the right half controls the left motor side. There exist an important connection between the two hemispheres (the *corpus callosum*) where information is exchanged.

The central concept of the JANUS architecture is the notion of *self-assessment* or self-supervision, within a hierarchy of adaptable network modules. The modules can modify themselves, and higher levels can act on other levels. In order that this might be possible, each module tries to estimate its limitations through self-assessment measures like confidence and reliability.

The JANUS project run from 1991 till 1997. It had to be abandoned because of lack of funding. The research progress was promising, but in 1997 JANUS was still far away to be used in a real application.The research has been published in the series GMD reports. The reports are out of print. The easiest access is via the WEB (http://citeseer.ist.psu.edu) or www.iais.fraunhofer.de/muehlenbein.html.

The low-level neural network architecture of JANUS has been investigated separately. We called it the PANDEMONIUM or MINOS architecture. Pandemonium had been originally proposed in 1958 by Selfridge [18]. The idea is to divide adaptively a complex domain, through the use of specialized agents, working in parallel. All these agents, or daemons in Selfridge's words, process the same signal in parallel, and each provides an answer with a certain confidence. The daemon with the largest confidence will be chosen for classification.

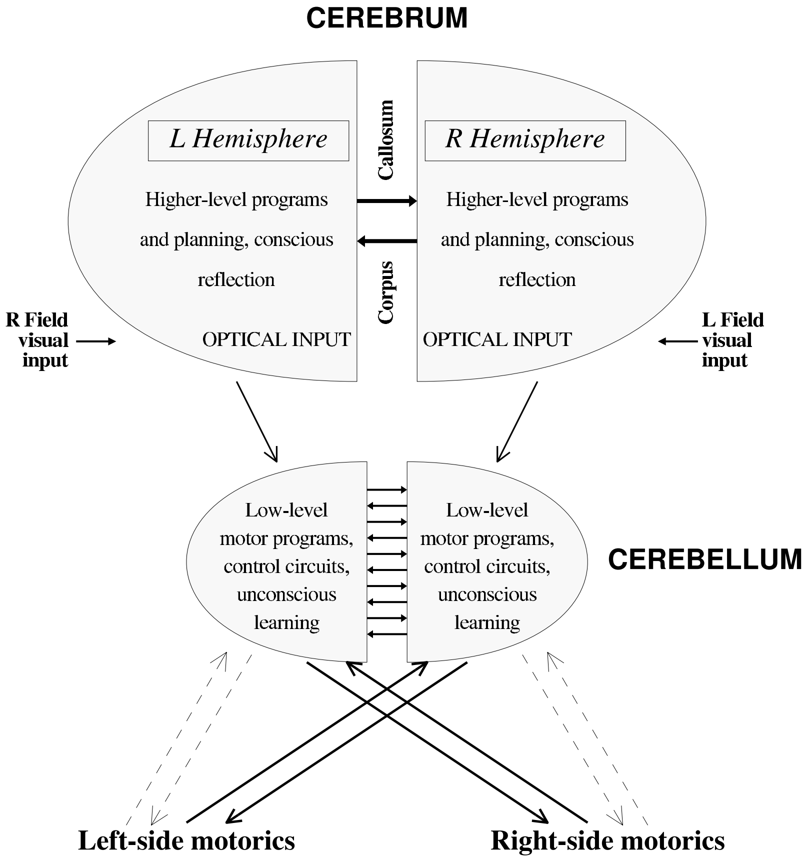

CEREBRUM
L Hemisphere
Higher-level programs and planning, conscious reflection
R Hemisphere
Higher-level programs and planning, conscious reflection
Callosum
Corpus
R Field visual input
OPTICAL INPUT
OPTICAL INPUT
L Field visual input
Low-level motor programs, control circuits, unconscious learning
Low-level motor programs, control circuits, unconscious learning
CEREBELLUM
Left-side motorics
Right-side motorics

Fig. 1 The architecture of the JANUS brain

Thus for a letter classification problem we might have 26 agents, each of which is specialized in recognizing a particular letter in all distortions. Each agent uses a number of filters. The *learning* method used by Selfridge was gradient descent for adapting the weights for each filter used.

We have taken this general idea and extended it to a modular system of neural networks. The central new idea is self-assessment by *reflection*. Each module observes its own behavior and produces information relating to the quality of its classification. The architecture was very successful in a number of classification tasks, but in the course of developing it more and more refinements had to be implemented. The interested reader is referred to [21, 1, 2].

9 Conclusion

Today computational intelligence is divided into many fields e.g. evolutionary computation, neural networks, fuzzy logic. These are further separated in a myriad of specialized techniques. In this paper I have recalled the fundamental research issues of machine intelligence by discussing the research of Alan Turing and John von Neumann. They represent two positions popular till today. For Turing the creation of machines with human-like intelligence was just a question of programming time. He estimated that sixty programmers had to work for fifty years. John von Neumann was more cautious. Using the example of visual analogy he doubted that human-like intelligent machines could be programmed in finite time and space. This lead him to the question if intelligent programs could automatically evolve by simulating evolution. While von Neumann solved the problem of self-reproducing automata, automata solving complex problems could not be yet obtained. I have identified the major problem of this approach: the programming of the environment seems to be as difficult as programming the human problem solving capabilities.

In my opinion it is not yet clear if Turing will be ultimately right that automata with human like intelligence could be programmed. Up to now computational intelligence was successful in specialized applications only, automata passing the Turing test or understanding languages are not yet in sight.

References

1. Beyer, U., Smieja, F.J.: Data exploration with reflective adaptive models. Computat. Statistics and Data Analysis 22, 193–211 (1996)
2. Beyer, U., Smieja, F.J.: Learning from examples, agent teams and the concept of reflection. International Journal of Pattern Recognition and Artificial Intelligence 10, 251–272 (1996)
3. Brooks, R.: From earwigs to humans. Robotics and Autonomous Systems 20, 291–304 (1997)
4. Brooks, R., Brezeal, C., Marjanovic, M., Scasselati, B., Williamson, M.: The Cog project: Building a humanoid robot. In: Nehaniv, C.L. (ed.) CMAA 1998. LNCS (LNAI), vol. 1562, pp. 52–87. Springer, Heidelberg (1999)
5. Burns, A.W.: Essays on Cellular Automata. University of Illinois Press, Urbana (1970)
6. Panton, K., Matuszek, C., Lenat, D., Schneider, D., Witbrock, M., Siegel, N., Shepard, B.: Common sense reasoning – from cyc to intelligent assistant. In: Cai, Y., Abascal, J. (eds.) Ambient Intelligence in Everyday Life. LNCS, vol. 3864, pp. 1–31. Springer, Heidelberg (2006)
7. Eccles, J.C.: The Human Mystery. Springer, New York (1979)
8. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco (1979)
9. Holland, J.H.: Iterative circuit computers. In: Essays on Cellular Automata [5], pp. 277–296
10. Holland, J.H.: Outline for a logical theory of adaptive systems. In: Essays on Cellular Automata, [5], pp. 296–319.
11. Holland, J.H.: Adaptation in Natural and Artificial Systems. Univ. of Michigan Press, Ann Arbor (1975/1992)
12. Lenat, D.B.: Cyc: A large-scale investment in knowledge infrastructure. Comm. ACM 38, 33–38 (1995)
13. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent un nervous activity. Bull. of Mathematical Biophysics 5, 115–137 (1943)

14. McMullin, B.: John von Neumann and the evolutionary growth of complexity: Looking backward, looking forward…. Artificial Life 6, 347–361 (2001)
15. Mühlenbein, H.: Towards a theory of organisms and evolving automata. In: Menon, A. (ed.) Frontiers of Evolutionary Computation, pp. 1–36. Kluwer Academic Publishers, Boston (2004)
16. Newell, A., Shaw, J.C., Simon, H.: Empirical explorations with the logic theory machine. In: Proc. Western Joint Computer Conference, vol. 11, pp. 218–239 (1957)
17. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufman, San Mateo (1988)
18. Selfridge, O.G.: Pandemonium: a paradigm for learning. In: Mechanisation of Thought Processes, pp. 511–529. Her Majesty's Stationery Office, London (1959)
19. Simon, H., Newell, A.: Heuristic problem solving: The next advance in operations research. Operations Research 6, 1–10 (1958)
20. Simon, H.A.: Models of my Life. MIT Press, Boston (1991)
21. Smieja, F.J.: The pandemonium system of reflective agents. IEEE Transact. on Neural Networks 7, 193–211 (1996)
22. Springer, S., Deutsch, G.: Left brain, right brain. W.H. Freeman, New York (1985)
23. Turing, A.M.: Computing machinery and intelligence. Mind 59, 433–460 (1950)
24. Turing, A.M.: Intelligent machinery. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence 6, pp. 3–23. Oxford University Press, Oxford (1969)
25. von Neumann, J.: The general and logical theory of automata. In: The world of mathematics, pp. 2070–2101. Simon and Schuster, New York (1954)
26. von Neumann, J.: Probabilistic logics and the synthesis of reliable organs from unreliable components. In: Annals of Mathematics Studies, vol. 34, pp. 43–99. Princeton University Press, Princeton (1956)
27. von Neumann, J.: Theory of Self-Reproducing Automata. University of Illinois Press, Urbana (1966)

Part II
Fusing Evolutionary Algorithms and Fuzzy Logic

Multiobjective Evolutionary Algorithms for Electric Power Dispatch Problem

Mohammad A. Abido

Abstract. The potential of Multiobjective Evolutionary Algorithms (MOEA) for solving a real-world power system multiobjective nonlinear optimization problem is comprehensively presented and discussed. In this work, the Non-dominated Sorting Genetic Algorithm (NSGA), Niched Pareto Genetic Algorithm (NPGA), and Strength Pareto Evolutionary Algorithm (SPEA) have been developed and successfully applied to the Environmental/Economic electric power Dispatch (EED) problem. These multiobjective evolutionary algorithms have been individually examined and applied to a standard test system. A hierarchical clustering algorithm is imposed to provide the power system operator with a representative and manageable Pareto set. Moreover, a fuzzy set theory based approach is developed to extract one of the Pareto-optimal solutions as the best compromise solution. Several optimization runs have been carried out on different cases of problem complexity. The results of the MOEA have been compared to those reported in the literature. The results confirm the potential and effectiveness of MOEA compared to the traditional multiobjective optimization techniques. In addition, the performance of MOEA have been assessed and evaluated using different measures of diversity, distribution, and quality of the obtained non-dominated solutions.

1 Introduction

Generally, the basic objective of the traditional economic dispatch (ED) of electric power generation is to schedule the committed generating unit outputs so as to meet the load demand at minimum operating cost while satisfying all generator and system equality and inequality constraints. This makes the ED problem a large-scale highly constrained nonlinear optimization problem.

Mohammad A. Abido
Electrical Engineering Department, King Fahd University of Petroleum & Minerals Dhahran 31261, Saudi Arabia
e-mail: mabido@kfupm.edu.sa

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 47–82.
springerlink.com

However, thermal power plants are major causes of atmospheric pollution because of the high concentration of pollutants they cause such as sulphur oxides SO_x and nitrogen oxides NO_x. Nowadays, the pollution minimization problem has attracted a lot of attention due to the public demand for clean air. In addition, the increasing public awareness of the environmental protection and the passage of the U.S. Clean Air Act Amendments of 1990 have forced the power utilities to modify their design or operational strategies to reduce pollution and atmospheric emissions of the thermal power plants [17, 24, 43].

Several strategies to reduce the atmospheric emissions have been proposed and discussed in the literature [43]. These include

- Installation of pollutant cleaning equipment such as gas scrubbers and electrostatic precipitators;
- Switching to low emission fuels;
- Replacement of the aged fuel-burners and generator units with cleaner and more efficient ones;
- Emission dispatching.

The first three options require installation of new equipment and/or modification of the existing ones that involve considerable capital outlay and, hence, they can be considered as long-term options. The emission dispatching option is an attractive short-term alternative in which the emission, in addition to the fuel cost objective, is to be minimized. In recent years, this option has received much attention [8, 10, 16, 18, 23] since it requires only a small modification of the basic economic dispatch to include emissions. Thus, the power dispatch problem can be handled as a multiobjective optimization problem with non-commensurable and contradictory objectives, since the optimum solution of the economic power dispatch problem is not environmentally the best solution.

Generally speaking, there are three approaches to solve the environmental/economic dispatch (EED) problem. The *first* approach treats the emission as a constraint with a permissible limit. The *second* approach treats the emission as another objective in addition to the usual cost objective, and the problem is converted to a single objective problem either by linear combination of both objectives or by considering one objective at a time for optimization. The *third* and the most recent approach handles both fuel cost and emission simultaneously as competing objectives.

In [8, 23] the problem has been reduced to a single objective problem by treating the emission as a constraint with a permissible limit. This formulation, however, has severe difficulty in getting the trade-off relations between cost and emission.

Alternatively, minimizing the emission has been handled as another objective in addition to the usual cost objective. A linear programming based optimization procedures in which the objectives are considered one at a time was presented in [18]. Unfortunately, this approach does not give any information regarding the trade-offs involved. In another research direction, the multiobjective EED problem was converted to a single objective problem by linear combination of the different objectives as a weighted sum [9, 10, 16]. The important aspect of this weighted sum method is that a set of non-inferior solutions can be obtained by varying the weights. Unfortunately, this requires multiple runs. Furthermore, this method cannot be used to find Pareto-optimal solutions in problems having a non-convex Pareto-optimal front. To avoid this difficulty, the ε-constraint method for multiobjective optimization was presented in [7, 45]. This method is based on optimization of the most preferred objective and considering the other objectives as constraints bounded by some allowable levels ε. The obvious weaknesses of this approach are that it is time-consuming and tends to find weakly non-dominated solutions.

The recent direction is to handle both objectives simultaneously as competing objectives. A fuzzy multiobjective optimization technique for the EED problem was proposed [41]. However, the solutions produced are sub-optimal and the algorithm does not provide a systematic framework for directing the search towards the Pareto-optimal front. A fuzzy satisfaction-maximizing decision approach was successfully applied to solve the bi-objective EED problem [27, 42]. However, extension of the approach to include more objectives such as security and reliability is a very involved question. A multiobjective stochastic search technique for the multiobjective EED problem was proposed in [14]. However, the technique is computationally involved and time-consuming. In addition, the genetic drift and search bias are severe problems that result in premature convergence. Therefore, additional efforts should be made to preserve the diversity of the non-dominated solutions.

In dealing with multiobjective optimization problems, classical search and optimization methods are not efficient for the following reasons.

- Most of them cannot find multiple solutions in a single run, thereby requiring them to be applied as many times as the number of desired Pareto-optimal solutions.
- Multiple applications of these methods do not guarantee finding widely different Pareto-optimal solutions.
- Most of them cannot efficiently handle problems with discrete variables and problems having multiple optimal solutions.
- Some algorithms are sensitive to the shape of the trade-off curve and cannot be used in problems having a non-convex Pareto-optimal front.

On the contrary, the studies on evolutionary algorithms have shown that these methods can be efficiently used to solve multiobjective optimization problems and eliminate most of the above difficulties of classical methods [11, 12, 13, 15, 20, 22, 26, 29, 31, 33, 34, 37, 39, 40, 44, 47, 49]. Since they use a population of solutions in their search, multiple Pareto trade-off solutions can be found in a single run.

Recently, different multiobjective evolutionary algorithms (MOEA) have been implemented and applied to the EED problem with impressive success [1, 2, 3, 4, 5]. In this chapter, implementations of different MOEA techniques to solve the real-world multiobjective EED problem have been carried out to assess their potential and effectiveness. Specifically speaking, Non-dominated Sorting Genetic Algorithm (NSGA) [40], Niched Pareto Genetic Algorithm (NPGA) [26], and Strength Pareto Evolutionary Algorithm (SPEA) [49] have been developed and implemented. It is worth mentioning that this work presents an exploratory study, aiming to demonstrate the potential of MOEA for solving the problem under consideration. The EED problem is formulated as a nonlinear constrained multiobjective optimization problem where fuel cost and environmental impact are treated as competing objectives. The potential of MOEA to handle this problem is investigated and discussed. A hierarchical clustering technique is implemented to provide the system operator with a representative and manageable Pareto trade-off set. In addition, a fuzzy-based mechanism is employed to extract the best compromise solution. Different cases with different complexities have been considered in this study. The MOEA techniques have been applied to the standard IEEE 30-bus 6-generator test system. These techniques were compared to each other and to classical multiobjective optimization techniques as well. The effectiveness of MOEA to solve the EED problem is demonstrated. The quality and diversity of the non-dominated solutions obtained by different MOEA techniques have been measured and assessed quantitatively.

2 EED Problem Formulation

The environmental/economic dispatch problem is to minimize two competing objective functions, fuel cost and emission, while satisfying several equality and inequality constraints. Generally the problem is formulated as follows [28].

2.1 Problem Objectives

Minimization of Fuel Cost: The generator cost curves are represented by quadratic functions and the total fuel cost $F(P_G)$ in ($/h) can be expressed as

$$F(P_G) = \sum_{i=1}^{N} a_i + b_i P_{G_i} + c_i P_{G_i}^2, \tag{1}$$

where N is the number of generators, a_i, b_i, and c_i are the cost coefficients of the i^{th} generator, and P_{Gi} is the real power output of thei^{th} generator. P_G is the vector of real power outputs of generators and is defined as

$$P_G = [P_{G_1}, P_{G_2}, \ldots, P_{G_N}]^T. \tag{2}$$

Minimization of Emission: The total emission $E(P_G)$ in (ton/h) of atmospheric pollutants such as sulphur oxides SO_x and nitrogen oxides NO_x caused by the operation of fossil-fueled thermal generation can be expressed as

$$E(P_G) = \sum_{i=1}^{N} 10^{-2}(\alpha_i + \beta_i P_{G_i} + \gamma_i P_{G_i}^2) + \zeta_i \exp(\lambda_i P_{G_i}), \tag{3}$$

where α_i, β_i, γ_i, ζ_i, and λ_i are coefficients of the i^{th} generator emission characteristics.

2.2 Problem Constraints

Generation capacity constraint: For stable operation, the real power output of each generator is restricted by lower and upper limits as follows:

$$P_{G_i}^{\min} \leq P_{G_i} \leq P_{G_i}^{\max}, \;\; i = 1,...,N. \tag{4}$$

Power balance constraint: the total electric power generation must cover the total electric power demand P_D and the real power loss in transmission lines P_{loss}. Hence,

$$\sum_{i=1}^{N} P_{G_i} - P_D - P_{loss} = 0. \tag{5}$$

Calculation of P_{loss} implies solving the load flow problem which has equality constraints on real and reactive power at each bus as follows

$$P_{G_i} - P_{D_i} - V_i \sum_{j=1}^{NB} V_j [G_{ij} \cos(\delta_i - \delta_j) + B_{ij} \sin(\delta_i - \delta_j)] = 0, \tag{6}$$

$$Q_{G_i} - Q_{D_i} - V_i \sum_{j=1}^{NB} V_j [G_{ij} \sin(\delta_i - \delta_j) - B_{ij} \cos(\delta_i - \delta_j)] = 0, \tag{7}$$

where i =1,2,...,NB; NB is the number of buses; Q_{Gi} is the reactive power generated ati^{th} bus; P_{Di} and Q_{Di} are the i^{th} bus load real and reactive power respectively; G_{ij} and B_{ij} are the transfer conductance and susceptance between bus i and bus j respectively; V_i and V_j are the voltage magnitudes at bus i and bus j respectively; δ_i and δ_j are the voltage angles at bus i and bus j respectively. The equality constraints in Equations (6) and (7) are nonlinear equations that can be solved using the Newton-Raphson method to generate a solution of the load flow problem. During the course of solution, the real power output of one generator, called the slack generator, is left to cover the real power loss and satisfy the equality constraint in (5). The load flow solution gives all bus voltage magnitudes and angles. Then, the real power loss in transmission lines can be calculated as

$$P_{loss} = \sum_{k=1}^{NL} g_k \left[V_i^2 + V_j^2 - 2V_iV_j \cos(\delta_i - \delta_j) \right], \tag{8}$$

where NL is the number of transmission lines; g_k is the conductance of the k^{th} line that connects bus i to bus j.

Security constraints: for secure operation, the apparent power flow through the transmission line S_l is restricted by its upper limit as follows:

$$S_{l_k} \leq S_{l_k}^{\max}, \;\; k = 1,...,NL. \tag{9}$$

It is worth mentioning that the k^{th} transmission line flow connecting bus i to bus j can be calculated as

$$S_{l_k} = (V_i \angle \delta_i) I_{ij}^{*}, \tag{10}$$

where I_{ij} is the current flow from bus i to bus j and it can be calculated as

$$I_{ij} = (V_i \angle \delta_i) \times \left[(V_i \angle \delta_i - V_j \angle \delta_j) \times y_{ij} + (V_i \angle \delta_i) \times j\frac{y}{2} \right], \tag{11}$$

where y_{ij} is the line admittance while y is the shunt susceptance of the line.

2.3 Problem Formulation

Aggregating the objectives and constraints, the problem can be mathematically formulated as a multiobjective optimization problem as follows.

$$\underset{P_G}{Minimize} \left[F(P_G), E(P_G) \right], \tag{12}$$

Subject to:

$$g(P_G) = 0, \tag{13}$$

$$h(P_G) \leq 0, \tag{14}$$

where g is the equality constraint representing the power balance while h are the inequality constraints representing the generation capacity and power system security.

3 Multiobjective Optimization

3.1 Principles and Definitions

Many real-world problems involve the simultaneous optimization of several objective functions. Generally, these functions are non-commensurable and often conflicting objectives. Multiobjective optimization with such conflicting objective functions gives rise to a set of optimal solutions, instead of a single optimum. The reason why many optimal solutions are obtained is that no one can be considered to be better than any other with respect to all objective functions. These optimal solutions are known as *Pareto-optimal* solutions.

A general multiobjective optimization problem consists of a number of objectives to be optimized simultaneously and is associated with a number of equality and inequality constraints. It can be formulated as follows:

$$\text{Minimize } f_i(x) \quad i = 1, \ldots, N_{obj}, \tag{15}$$

$$\text{Subject to}: \begin{cases} g_j(x) = 0 & j = 1, \ldots, M, \\ h_k(x) \leq 0 & k = 1, \ldots, K, \end{cases} \tag{16}$$

where f_i is the i^{th} objective function, x is a candidate solution, and N_{obj} is the number of objectives.

For a multiobjective optimization problem, any two solutions x_1 and x_2 can have one of two relationships - one dominates the other or none dominates the other. In a minimization problem, without loss of generality, a solution x_1 dominates x_2 iff the following two conditions are satisfied:

$$1. \quad \forall i \in \{1, 2, \ldots, N_{obj}\} : f_i(x_1) \leq f_i(x_2), \tag{17}$$

$$2. \quad \exists j \in \{1, 2, \ldots, N_{obj}\} : f_j(x_1) < f_j(x_2). \tag{18}$$

If any of the above conditions is violated, the solution x_1 does not dominate the solution x_2. If x_1 dominates the solution x_2, x_1 is called the non-dominated solution within the set $\{x_1, x_2\}$. The solutions that are non-dominated within the entire search space are denoted as *Pareto-optimal* and constitute the *Pareto-optimal set*. The objective function values associated with the non-dominated solutions in Pareto-optimal set comprise the *Pareto-optimal front*.

3.2 Fitness Assignment

Fonseca and Fleming [22] categorized several MOEAs and compared different fitness assignment approaches. They classified these approaches as aggregating approaches, non-Pareto-based approaches, and Pareto-based approaches.

Aggregating approaches combine the problem objectives into a single function that is used for fitness calculation. Although these approaches have the advantage of producing one single solution, they require well-known domain knowledge that is often not available. In addition, multiple runs are required to find a family of non-dominated solutions and to identify the a Pareto trade-off front. The most popular aggregating approaches are the weighted-sum, goal programming, and ε-constrained methods [11].

To overcome the difficulties involved in the aggregating approaches, alternative techniques based on population policies, selection criteria, or special handling of the objectives have been developed. These approaches are known as non-Pareto-based approaches. The advantage of these approaches is that multiple non-dominated solutions can be simultaneously evolved in a single run. These approaches, however, are often sensitive to the non-convexity of Pareto-optimal sets. The most popular non-Pareto-based approaches are the Vector Evaluated Genetic Algorithm (VEGA) [37], multi-sexual genetic algorithm [33], and weighted Min-Max approach [12].

The basic idea of the Pareto-based fitness assignment is to find a set of solutions in the population that are non-dominated by the rest of the population. These solutions are then assigned the highest rank and eliminated from further contention. Generally, all approaches of this class explicitly use Pareto dominance in order to determine the reproduction probability of each individual. Some Pareto-based approaches are NSGA, NPGA, and SPEA.

3.3 Diversity Preservation

In general, the goal of a multiobjective optimization algorithm is not only to guide the search towards the Pareto-optimal front but also to maintain population diversity in the trade-off front. Unfortunately, a simple evolutionary algorithm tends to converge towards a single solution due to selection pressure, selection noise, and operator disruption [34]. Several approaches have been developed in order to overcome this problem, preserve the diversity in the population, and prevent premature convergence. These approaches are classified as niching techniques and non-niching techniques. Niching algorithms are characterized by their capabilities of maintaining stable subpopulations (niches).

Fitness sharing is the most frequently used niching technique. The basic idea behind this technique is: the more individuals are located in the neighborhood of a certain individual, the more its fitness value is degraded. The neighborhood is defined in terms of a distance measure d_{ij} and specified by the niche radius σ_{share}.

Restricted mating is the most frequently used non-niching technique. In this technique, two individuals are allowed to mate only if they are within a certain distance. This mechanism may avoid the formation of lethal individuals and therefore improve the online performance. However, it does not appear to be widely used in the field of multiobjective evolutionary algorithms [22].

4 Multiobjective Evolutionary Algorithms

The recent studies on evolutionary algorithms have shown that these methods can be efficiently used to eliminate most of the difficulties of the classical optimization methods. In this study, the basic Pareto-based MOEA have been developed and implemented. Specifically speaking, NSGA [40], NPGA [26], and SPEA [49] have been considered in this work.

4.1 Non-dominated Sorted Genetic Algorithm (NSGA)

Srinivas and Deb [40] developed NSGA in which a ranking selection method is used to emphasize current non-dominated solutions and a niching method is used to maintain diversity in the population. Before the selection is performed, the population is first ranked in several steps. At first, the non-dominated solutions in the population are identified. These non-dominated solutions constitute the first non-dominated front and are assigned the same dummy fitness value. To maintain diversity in the population, these non-dominated solutions are then shared with their dummy fitness values. Phenotypic sharing on the decision space is used in this technique. After sharing, these non-dominated individuals are ignored temporarily to process the rest of population members. The above procedure is repeated to find the second level of non-dominated solutions in the population. Once they are identified, a dummy fitness value, which is a little smaller than the worst shared fitness value observed in solutions of first non-dominated set, is assigned. Thereafter, the sharing procedure is performed among the solutions of the second non-domination level and the shared fitness values are found as before. This process is continued until all population members are assigned a shared fitness value. The population is then reproduced with the shared fitness values. A stochastic remainder selection is used in this study.

In the first generation, the non-dominated solutions of the first front are stored in an external set. After ranking in the subsequent generations, this external set is extended by adding the solutions from the new first fronts, and removing any dominated solutions. Generally, NSGA includes two main steps: fitness assignment and fitness sharing.

Fitness assignment: the basic idea of this approach is to find a set of solutions in the population that are non-dominated by the rest of the population. Consider a set of N population members, each having N_{obj} objective function values, the following procedure is used to find the nondominated set of solutions:-

Step 1: Initiate the individual counter i with $i = 1$.
Step 2: For all $j = 1,\ldots,N$ and $j \neq i$, compare solutions x^i and x^j for domination using the conditions of domination.
Step 3: If for any j, x^i is dominated by x^j, mark x^i as dominated.
Step 4: If all individuals in the population are considered, Go to Step 5, else set $i = i+1$ and go to Step 2.
Step 5: All solutions that are not marked dominated are non-dominated solutions.

These solutions represent the first front and are eliminated from further contention. This process continues until the population is properly ranked.

Fitness sharing: the basic idea behind sharing is: the more individuals are located in the neighborhood of a certain individual, the more its fitness value is degraded. The neighborhood is defined in terms of a distance measure d and specified by the niche radius σ_{share}. Given a set of n_k solutions in the kth front each having a dummy fitness value f_k , the sharing procedure is performed in the following way [26] for each solution $i = 1,\ldots,n_k$:-

Step 1: Compute a normalized Euclidean distance measure with another solution j in the kth nondominated front, as follows:

$$d_{ij} = \sqrt{\sum_{k=1}^{P} \left(\frac{x_k^i - x_k^j}{x_k^u - x_k^l} \right)^2} \tag{19}$$

where P is the number of variables in the problem. x_k^u and x_k^l are the upper and lower bounds of variable x_k.

Step 2: This distance d_{ij} is compared with a prespecified parameter σ_{share} and the following sharing function value is computed:

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}} \right)^2, & \text{if } d_{ij} \le \sigma_{share} \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

Step 3: Increment j. If $j \le n_k$, go to Step 1 else calculate niche count for ith solution as follows:

$$m_i = \sum_{j=1}^{n_k} Sh(d_{ij}) \tag{21}$$

Step 4: Degrade the dummy fitness f_k of ith solution in the kth nondomination front to calculate the shared fitness, f_i^*, as follows:

$$f_i^* = \frac{f_k}{m_i} \tag{22}$$

This procedure is continued for all $i = 1,\ldots,n_k$ and a corresponding f_i^* is found. Thereafter, the smallest value f_k^{min} of all f_i^* in the kth nondominated front is found for further processing. The dummy fitness of the next non-dominated front is assigned to be $f_{k+1} = f_k^{\min} - \varepsilon_k$, where ε_k is a small positive number.

4.2 Niched Pareto Genetic Algorithm (NPGA)

Horn et al [26] proposed a tournament selection scheme based on Pareto dominance. Two competing individuals and a comparison set of other individuals are picked at random from the population. The number of individuals in the comparison set is given by the parameter t_{dom}. Generally, the tournament selection is carried out as follows. If one candidate is dominated by one or more members of the comparison set while the other is not, then the later will be selected for reproduction. If neither or both candidates are dominated by any members of the comparison set, then the winner will be decided by sharing. The phenotypic sharing on the attribute space is used in this technique. Generally, the tournament selection and sharing procedure are carried out as follows.

Pareto domination tournaments: Consider a set of N population members, each having N_{obj} objective function values. The following procedure can be used to find the non-dominated set of solutions:

Step 1: Begin with $i = 1$.
Step 2: Randomly pick two candidates for selection x_1 and x_2.
Step 3: Randomly pick a comparison set of individuals from the population.
Step 4: Compare each candidate, x_1 and x_2, against each individual in the comparison set for domination using the conditions for domination.
Step 5: If one candidate is dominated by the comparison set while the other is not, then select the latter for reproduction and go to Step 7, else proceed to Step 6.
Step 6: If neither or both candidates are dominated by the comparison set, then use sharing to choose the winner.
Step 7: If $i = N$ is reached, stop selection procedure, else set $i = i + 1$ and Go to Step 2.

Sharing procedure: To prevent the genetic drift problem, a form of sharing should be carried out when there is no preference between two candidates. This form of sharing maintains the genetic diversity along the population fronts and allows the GA to develop a reasonable representation of the Pareto-optimal front. Generally, the basic idea behind sharing is: the more individuals are located in the neighborhood of a certain individual, the more its fitness value is degraded. The sharing procedure is performed in the following way for the candidate i:-

Step 1: Begin with j = 1.
Step 2: Compute a normalized Euclidean distance measure with another individual$j$ in the current population, as follows:

$$d_{ij} = \sqrt{\sum_{k=1}^{N_{obj}} \left(\frac{J_k^i - J_k^j}{J_k^u - J_k^l} \right)^2} \tag{23}$$

where N_{obj} is the number of problem objectives. The parameters J_k^u and J_k^l are the upper and lower values of the k-th objective function J_k.
Step 3: This distance d_{ij} is compared with a prespecified niche radius σ_{share} and the following sharing function value is computed as:

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}} \right)^2, \text{ if } d_{ij} \leq \sigma_{share} \\ 0, \qquad\qquad \text{otherwise} \end{cases} \tag{24}$$

Step 4: Set $j = j{+}1$. If $j \leq N$, go to Step 2, else calculate niche count for the candidate i as follows:

$$m_i = \sum_{j=1}^{N} Sh(d_{ij}) \tag{25}$$

Step 5: Repeat the above steps for the second candidate.
Step 6: Compare m_1 and m_2. If $m_1 < m_2$, then choose the first candidate, else choose the second candidate.

4.3 Strength Pareto Evolutionary Algorithm (SPEA)

Zitzler and Thiele [49] presented SPEA as a potential algorithm for multiobjective optimization. This technique stores externally the individuals that represent a non-dominated front among all solutions considered so far. All individuals in the external set participate in selection. SPEA uses the concept of Pareto dominance in order to assign scalar fitness values to individuals in the current population. The procedure starts with assigning a real value s in [0,1) called the strength for each individual in the external set. The strength of an individual is proportional to the number of individuals covered by it. The strength of a Pareto solution is at the same time its fitness. Subsequently, the fitness of each individual in the population is the sum of the strengths of all external Pareto solutions by which it is covered. In order to guarantee that Pareto solutions are most likely to be produced, one is added to the resulting value. This fitness assignment ensures that the search is directed towards the non-dominated solutions and, in the same time, the diversity among dominated and non-dominated solutions is maintained.

The basic elements of the SPEA technique are briefly stated and defined as follows:-

- *External set*: - It is a set of non-dominated solutions. These solutions are stored externally and updated continuously. Ultimately, the solutions stored in this set represent the Pareto optimal front.
- *Strength of an individual*: - It is an assigned real value s [0,1) for each individual in the external set. The strength of an individual is proportional to the number of individuals covered by it.
- *Fitness of population individuals*: - The fitness of each individual in the population is the sum of the strengths of all external solutions by which it is covered. It is worth mentioning that, unlike the technique presented in [14], the fitness of a population member is determined only in relation to the individuals stored in the external set. This significantly reduces the computational burden of the fitness assignment process. In fact, the strength of an individual in the external set is at the same time its fitness.

Generally, SPEA can be described in the following steps.

Step 1 **(Initialization):** Generate an initial population and create the empty external set.

Step 2 **(External set updating):** The external set is updated as follows.

(a) Search the population for the non-dominated individuals and copy them to the external Pareto set.

(b) Search the external Pareto set for the non-dominated individuals and remove all dominated solutions from the set.

(c) If the number of the individuals externally stored in the Pareto set exceeds a prespecified maximum size, reduce the set by means of clustering.

Step 3 **(Fitness assignment):** Calculate the fitness values of individuals in both external Pareto set and the population as follows.

(a) Assign a strength *s* to each individual in the external set. The strength is proportional to the number of individuals covered by that individual.
(b) The fitness of each individual in the population is the sum of the strengths of all external Pareto solutions which dominate that individual. A small positive number is added to the resulting sum to guarantee that Pareto solutions are most likely to be produced.

Step 4 **(Selection):** Combine the population and the external set individuals. Select two individuals at random and compare their fitness. Select the better one and copy it to the mating pool. Repeat the selection process *N* times to fill the mating pool
Step 5 **(Crossover and Mutation):** Perform the crossover and mutation operations according to their probabilities to generate the new population.
Step 7 **(Termination):** Check for stopping criteria. If any one is satisfied *then* stop *else* copy new population to old population and go to Step 2. In this study, the search will be stopped if the generation counter exceeds its maximum number.

It is worth mentioning that new and revised versions of MOEA have been presented such as NSGA-II [15, 29], SPEA2 [47], and multiobjective particle swarm optimization MOPSO [13]. Recently, different studies in analysis, test cases, and applications of MOEA have also been discussed [20, 31, 39].

5 MOEA Implementation

5.1 Reducing the Pareto Set by Clustering

The Pareto-optimal set can be extremely large or even contain an infinite number of solutions. In this case, reducing the set of non-dominated solutions without destroying the characteristics of the trade-off front is desirable from the decision maker's point of view. An average linkage based hierarchical clustering algorithm [35] used by SPEA [49] is employed to reduce the Pareto set to a manageable size. It works iteratively by joining the adjacent clusters until the required number of groups is obtained. It can be described as:

> *Given a set P for which its size exceeds the maximum allowable size N, it is required to form a subset P^* with the size N*

The algorithm is illustrated in the following steps.

Step 1: Initialize cluster set C; each individual $i \in P$ constitutes a distinct cluster.
Step 2: If the number of clusters $\leq N$, then go to Step 5, else go to Step 3.

Step 3: Calculate the distance of all possible pairs of clusters. The distance d_c of two clusters c_1 and $c_2 \in C$ is given as the average distance between pairs of individuals across the two clusters

$$d_c = \frac{1}{n_1 . n_2} \sum_{i_1 \in c_1, i_2 \in c_2} d(i_1, i_2) \tag{26}$$

where n_1 and n_2 are the number of individuals in the clusters c_1 and c_2 respectively. The function d reflects the distance in the objective space between individuals i_1 and i_2.

Step 4: Determine two clusters with minimal distance d_c. Combine these clusters into a larger one. Go to Step 2.

Step 5: Find the centroid of each cluster. Select the nearest individual in this cluster to the centroid as a representative individual and remove all other individuals from the cluster.

Step 6: Compute the reduced non-dominated set P^* by uniting the representatives of the clusters.

5.2 Best Compromise Solution

Fuzzy set theory has been implemented to efficiently derive a candidate trade-off solution for the decision makers [19, 21, 36]. Upon having the final non-dominated set, the proposed approach presents a fuzzy-based mechanism to extract a single non-dominated solution from the trade-off front as the best compromise solution. Due to the imprecise nature of the decision maker's judgment, the ith objective function of a solution in the non-dominated set, F_i, is represented by a membership function μ_i defined as [36]

$$\mu_i = \begin{cases} 1, & F_i \le F_i^{\min}, \\ \frac{F_i^{\max} - F_i}{F_i^{\max} - F_i^{\min}}, & F_i^{\min} < F_i < F_i^{\max}, \\ 0, & F_i \ge F_i^{\max}. \end{cases} \tag{27}$$

where F_i^{max} and F_i^{min} are the maximum and minimum values of the ith objective function respectively. For each non-dominated solution k, the normalized membership function μ^k is calculated as

$$\mu^k = \frac{\sum\limits_{i=1}^{N_{obj}} \mu_i^k}{\sum\limits_{j=1}^{M} \sum\limits_{i=1}^{N_{obj}} \mu_i^j}, \tag{28}$$

where M is the number of non-dominated solutions. The best compromise solution is the one having the maximum of μ^k. As a matter of fact, arranging all solutions in the trade-off front in descending order according to their membership function will provide the decision maker with a priority list of non-dominated solutions. This will guide the decision maker in view of the current operating conditions.

5.3 Real-Coded Genetic Algorithm

Due to the difficulties of binary representation when dealing with a continuous search space of large dimensions, a real-coded genetic algorithm (RCGA) [25] has been implemented in this study. A decision variable x_i is represented by a real number which lies between a lower limit a_i and upper limit b_i, i.e. $x_i \in [a_i, b_i]$. The RCGA crossover and mutation operators are described as follows:-

Crossover: A blend crossover operator (BLX-α) has been employed in this study. This operator starts by choosing randomly a number from the interval $[x_i - \alpha(y_i - x_i), y_i + \alpha(y_i - x_i)]$, where x_i andy_i are the i^{th} parameter values of the parent solutions and$x_i < y_i$. In order to ensure the balance between exploitation and exploration of the search space, α = 0.5 is selected. This operator can be depicted as shown in Figure 1.

Mutation: The non-uniform mutation has been employed in this study. In this operator, the new value x_i' of the parameter x_i after mutation at generation t is given as

$$x_i' = \begin{cases} x_i + \Delta(t, b_i - x_i), & \text{if } \tau = 0, \\ x_i - \Delta(t, x_i - a_i), & \text{if } \tau = 1, \end{cases} \tag{29}$$

and;

$$\Delta(t,y) = y\left(1 - r^{(1-\frac{t}{g_{\max}})^{\beta}}\right), \tag{30}$$

where τ is a binary random number, r is a random number $r \in [0,1]$, g_{max} is the maximum number of generations, and β is a positive constant chosen arbitrarily. In

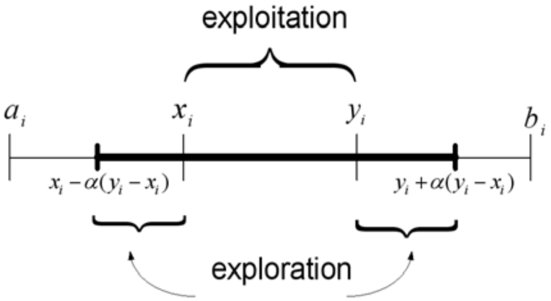


Fig. 1 Blend crossover operator (BLX-α)

this study, $\beta = 5$ was selected. This operator gives a value $x_i \in [a_i,b_i]$ such that the probability of returning a value close to x_i increases as the algorithm advances. This encourages uniform search in the initial stages when t is small, and local search in the later stages.

5.4 The Computational Flow

In this study, the basic MOEAs have been developed in order to make them suitable for solving real-world nonlinear constrained optimization problems. The following modifications have been incorporated in the basic algorithms.

(a) The constraint-handling approach adopted in this work is to restrict the search within the feasible region. Therefore, a procedure is imposed to check the feasibility of the initial population individuals and the children generated through GA operations. This ensures the feasibility of the non-dominated solutions.
(b) A procedure for updating the non-dominated archive set is developed. In every generation, the non-dominated solutions in the first front are combined with the existing archive set. The augmented set is processed to extract the non-dominated solutions that represent the updated non-dominated archive.
(c) A fuzzy-based mechanism is employed to extract the best compromise solution over the trade-off curve and assist the power system operator to adjust the generation levels efficiently.

The solution procedure starts with generating the initial population at random. A feasibility check procedure has been developed and superimposed on the MOEA to restrict the search to the feasible region. The objective functions are evaluated for each individual. The GA operations are applied and a new population is generated. This process is repeated until the maximum number of generations is reached. All techniques used in this study were implemented along with the above modifications using the FORTRAN language. The computational flow charts of the developed NSGA, NPGA, and SPEA are shown in Figures 2, 3, and 4 respectively.

5.5 Settings of the Proposed Approach

For all optimization runs, the population size was set at 200. The size of the external set was chosen as 25. If the number of the non-dominated archive exceeds this bound, the hierarchical clustering technique is called. Since the population in SPEA is augmented to include the externally stored set for selection process, the population size in SPEA was reduced to 175 individuals only. Crossover and mutation probabilities were chosen as 0.9 and 0.01 respectively in all optimization runs. Several runs have been carried out to set the parameters of each technique in order to get the best results for fair comparison.

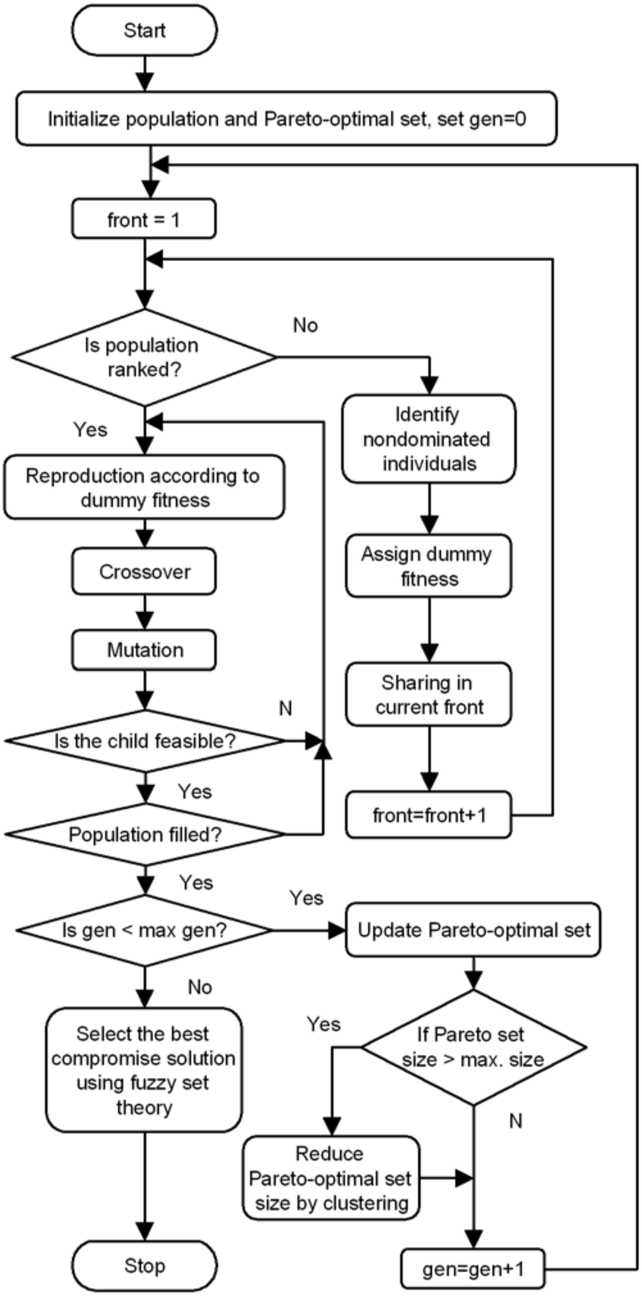

Start
Initialize population and Pareto-optimal set, set gen=0
front = 1
Is population ranked?
No
Yes
Identify nondominated individuals
Reproduction according to dummy fitness
Assign dummy fitness
Crossover
Mutation
Sharing in current front
N
Is the child feasible?
Yes
front=front+1
Population filled?
Yes
Yes
Is gen < max gen?
Update Pareto-optimal set
No
Yes
If Pareto set size > max. size
Select the best compromise solution using fuzzy set theory
N
Reduce Pareto-optimal set size by clustering
Stop
gen=gen+1

Fig. 2 Computational flow of the developed NSGA

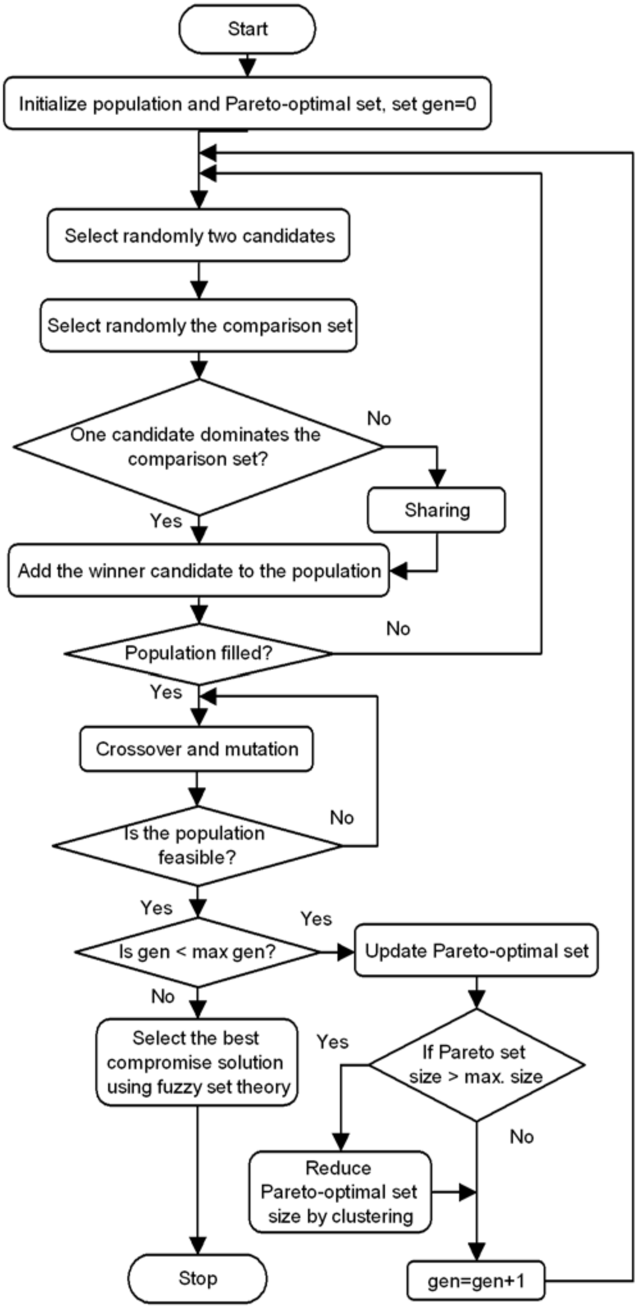

Start
Initialize population and Pareto-optimal set, set gen=0
Select randomly two candidates
Select randomly the comparison set
One candidate dominates the comparison set?
No
Sharing
Yes
Add the winner candidate to the population
No
Population filled?
Yes
Crossover and mutation
No
Is the population feasible?
Yes
Yes
Is gen < max gen?
Update Pareto-optimal set
No
Select the best compromise solution using fuzzy set theory
Yes
If Pareto set size > max. size
No
Reduce Pareto-optimal set size by clustering
Stop
gen=gen+1

Fig. 3 Computational flow of the developed NPGA

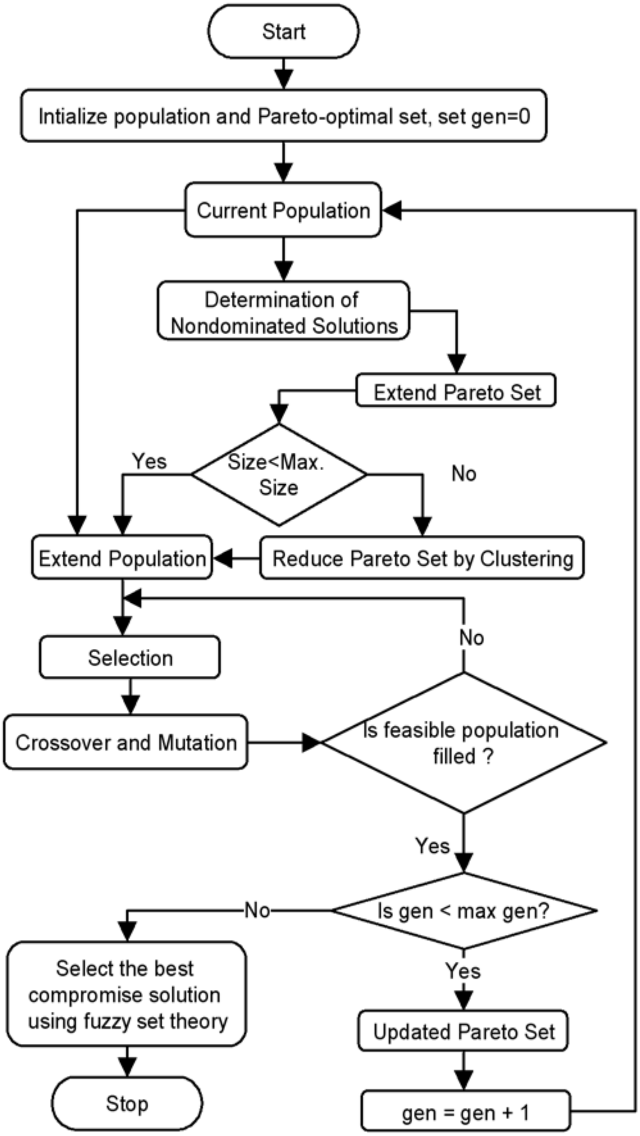

Start
Intialize population and Pareto-optimal set, set gen=0
Current Population
Determination of Nondominated Solutions
Extend Pareto Set
Size<Max. Size
Yes
No
Extend Population
Reduce Pareto Set by Clustering
No
Selection
Crossover and Mutation
Is feasible population filled ?
Yes
Is gen < max gen?
No
Yes
Select the best compromise solution using fuzzy set theory
Updated Pareto Set
Stop
gen = gen + 1

Fig. 4 Computational flow of the developed SPEA

6 Results and Discussions

In this study, the standard IEEE 6-generator 30-bus test system is considered to assess the potential of MOEAs for solving the EED problem. The power system considered has 30 buses or electrical nodes interconnected with each other with 41

Table 1 Generator Cost and Emission Coefficients

		G_1	G_2	G_3	G_4	G_5	G_6
Cost	a	10	10	20	10	20	10
	b	200	150	180	100	180	150
	c	100	120	40	60	40	100
Emission	α	4.091	2.543	4.258	5.326	4.258	6.131
	β	-5.554	-6.047	-5.094	-3.550	-5.094	-5.555
	γ	6.490	5.638	4.586	3.380	4.586	5.151
	ζ	2.0E-4	5.0E-4	1.0E-6	2.0E-3	1.0E-6	1.0E-5
	λ	2.857	3.333	8.000	2.000	8.000	6.667

Table 2 Problem complexity for the cases considered

	Equality Constraints	***Inequality Constraints***
Case 1	1	6
Case 2	60	6
Case 3	60	47

transmission lines. The system has also 6 generation plants to supply 23 electrical loads. The single-line diagram of this system is shown in Figure 5. The line data and bus data are given in the Appendix. The values of fuel cost and emission coefficients are given in Table 1.

To demonstrate the effectiveness of the MOEA, three different cases have been considered as follows:

Case 1: For the purpose of comparison with the reported results, the system is considered as lossless and the security constraint is released. Therefore, the problem constraints are the power balance constraint without P_{loss} and the generation capacity constraint.

Case 2: P_{loss} is considered in the power balance constraint and the generation capacity constraint is also considered.

Case 3: All constraints are considered.

For fair comparison among the developed techniques, 10 different optimization runs have been carried out in all cases considered. Table 2 shows the problem complexity with all cases in terms of the number of equality and inequality constraints.

At first, the fuel cost objective and emission objective are optimized individually to explore the extreme points of the trade-off surface in all cases. In this case, the standard GA has been implemented as the problem becomes a single objective optimization problem. The best results of cost and emission when optimized individually for all cases are given in Table 3.

***Case 1*:** NSGA, NPGA, and SPEA have been applied to the problem and both objectives were treated simultaneously as competing objectives. For NPGA, the niche radius was chosen based on the guidelines in [26] and the size of the comparison set

Table 3 Best solutions for cost and emission optimized

	***Case* 1**		***Case* 2**		***Case* 3**	
	Cost	*Emission*	*Cost*	*Emission*	*Cost*	*Emission*
P_{G1}	0.1095	0.4058	0.1152	0.4101	0.1475	0.4693
P_{G2}	0.2997	0.4592	0.3055	0.4631	0.3340	0.5223
P_{G3}	0.5245	0.5380	0.5972	0.5435	0.7864	0.6479
P_{G4}	1.0160	0.3830	0.9809	0.3895	1.0096	0.4734
P_{G5}	0.5247	0.5379	0.5142	0.5439	0.1072	0.1784
P_{G6}	0.3596	0.5101	0.3542	0.5150	0.4806	0.5761
Cost	**600.11**	638.26	**607.78**	645.22	**618.50**	654.14
Emission	0.2221	**0.1942**	0.2199	**0.1942**	0.2302	**0.2016**

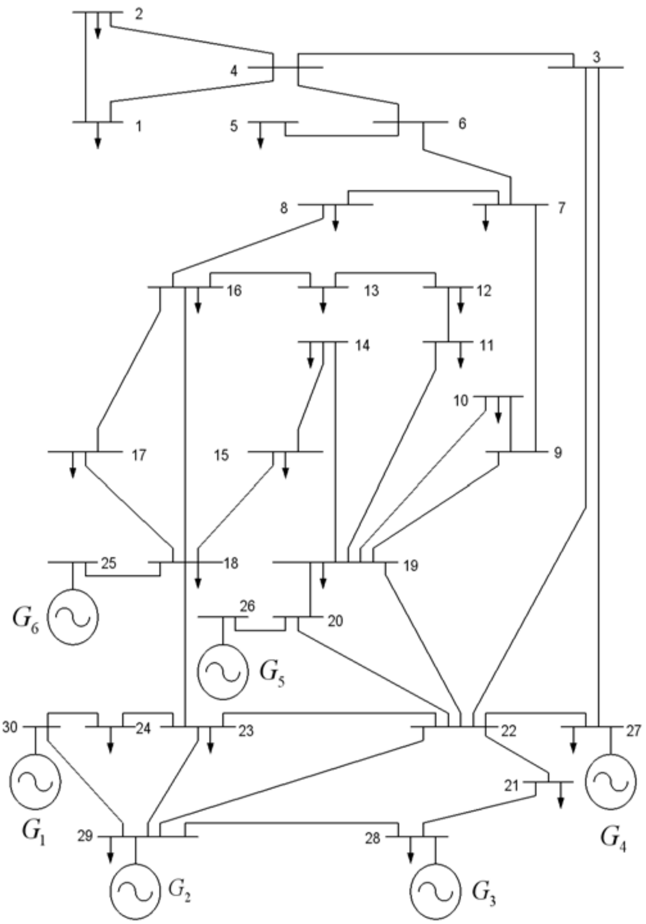


Fig. 5 Single-line diagram of the test system

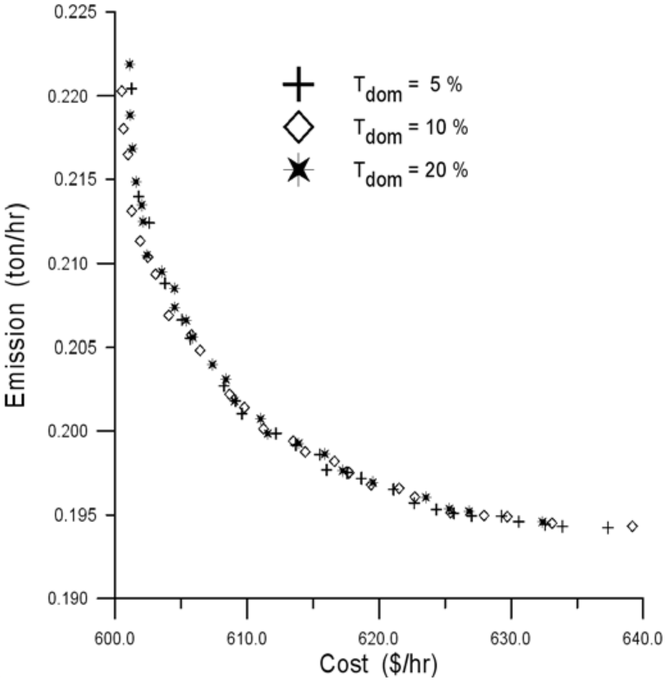


Fig. 6 NPGA with different settings of t_{dom} parameter

t_{dom} was determined experimentally. The algorithm was tested several times with different values for t_{dom} starting from 5% to 50% of the population size with a step of 5%. Only a part of the results is shown in Figure 6 for the purpose of clarity. Experimental results have shown a favorable value of t_{dom} at 10% for our problem instance, whereas the performance degrades for values t_{dom} greater than 20%. Therefore, t_{dom} is set at 10% of the population size.

The non-dominated fronts of all techniques for the best optimization runs are shown in Figure 7. It is clear that the non-dominated solutions have good diversity characteristics. It is quite clear that the problem is efficiently solved by these techniques. The results also show that SPEA has better diversity characteristics. The best cost and best emission solutions obtained out of 10 runs by different techniques are given in Table 4. It is clear that SPEA gives best cost and best emission compared to others.

The best results of the MOEAs were compared to those reported using linear programming (LP) [18] and a multiobjective stochastic search technique (MOSST) [14]. The comparison is shown in Table 5. It is quite evident that the MOEAs give better fuel cost results than the traditional methods, as a reduction more than 5 $/h is observed with less level of emission in case of SPEA. The results also confirm the potential of multiobjective evolutionary algorithms to solve real-world highly nonlinear constrained multiobjective optimization problems.

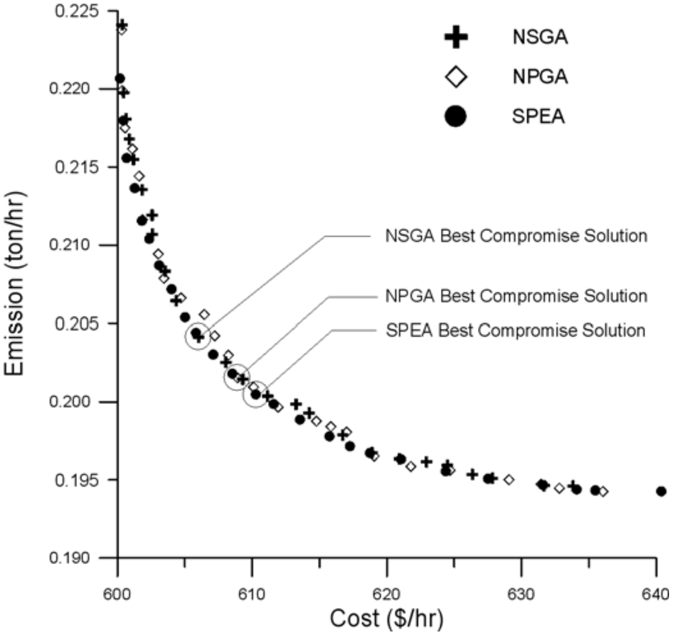


Fig. 7 Comparison of trade-off fronts, Case 1

Table 4 The best solutions out of 10 runs for cost and emission of MOEA, Case 1

	NSGA		*NPGA*		*SPEA*	
	Cost	*Emission*	*Cost*	*Emission*	*Cost*	*Emission*
P_{G1}	0.1038	0.4072	0.1116	0.4146	0.1009	0.4240
P_{G2}	0.3228	0.4536	0.3153	0.4419	0.3186	0.4577
P_{G3}	0.5123	0.4888	0.5419	0.5411	0.5400	0.5301
P_{G4}	1.0387	0.4302	1.0415	0.4067	0.9903	0.3721
P_{G5}	0.5324	0.5836	0.4726	0.5318	0.5336	0.5311
P_{G6}	0.3241	0.4707	0.3512	0.4979	0.3507	0.5190
Cost	**600.34**	633.83	**600.31**	636.04	**600.22**	640.42
Emission	0.2241	**0.1946**	0.2238	**0.1943**	0.2206	**0.1942**

Table 5 The best fuel cost and emission out of 10 runs of MOEA compared to traditional algorithms

	LP **[18]**	***MOSST*****[14]**	***NSGA***	***NPGA***	***SPEA***
Best Cost	606.31	605.89	600.34	600.31	600.22
Emission	0.2233	0.2222	0.2241	0.2238	0.2206
Best Emission	0.1942	0.1942	0.1946	0.1943	0.1942
Cost	639.60	644.11	633.83	636.04	640.42

Table 6 The best solutions out of 10 runs for cost and emission of MOEA, Case 2

	NSGA		*NPGA*		*SPEA*	
	Cost	*Emission*	*Cost*	*Emission*	*Cost*	*Emission*
P_{G1}	0.1447	0.3929	0.1425	0.4064	0.1279	0.4145
P_{G2}	0.3066	0.3937	0.2693	0.4876	0.3163	0.4450
P_{G3}	0.5493	0.5818	0.5908	0.5251	0.5803	0.5799
P_{G4}	0.9894	0.4316	0.9944	0.4085	0.9580	0.3847
P_{G5}	0.5244	0.5445	0.5315	0.5386	0.5258	0.5348
P_{G6}	0.3542	0.5192	0.3392	0.4992	0.3589	0.5051
Cost	**607.98**	638.98	**608.06**	644.23	**607.86**	644.77
Emission	0.2191	**0.1947**	0.2207	**0.1943**	0.2176	**0.1943**

***Case 2*:** With the problem complexity shown in Table 2, MOEA techniques have been implemented and compared. Figure 8 shows the trade-off fronts of different techniques for the best optimization runs. It is evident that the non-dominated solutions obtained have good diversity characteristics. The closeness of the non-dominated solutions of different techniques demonstrates good performance characteristics of MOEAs. The best solutions obtained out of 10 runs by different techniques are given in Table 6.

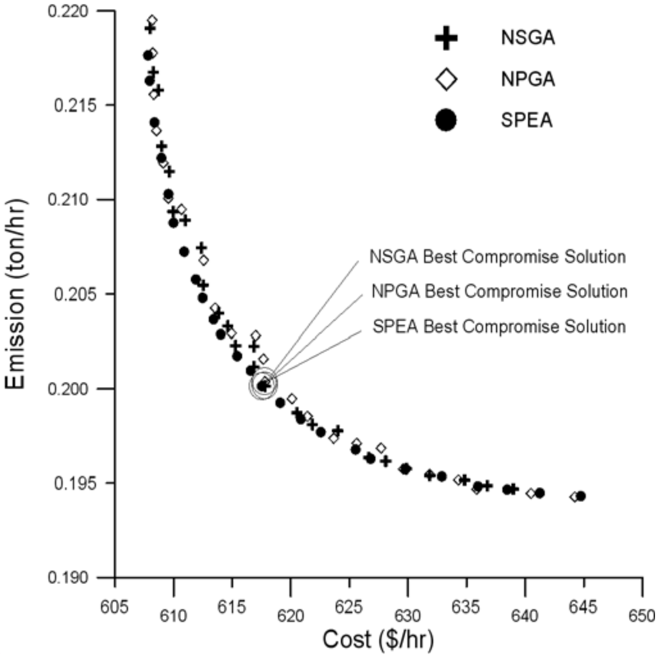


Fig. 8 Comparison of trade-off fronts, Case 2

Table 7 The best solutions out of 10 runs for cost and emission of MOEA, Case 3

	NSGA		***NPGA***		***SPEA***	
	Cost	*Emission*	*Cost*	*Emission*	*Cost*	*Emission*
P_{G1}	0.1358	0.4403	0.1127	0.4753	0.1319	0.4419
P_{G2}	0.3151	0.4940	0.3747	0.5162	0.3654	0.4598
P_{G3}	0.8418	0.7509	0.8057	0.6513	0.7791	0.6944
P_{G4}	1.0431	0.5060	0.9031	0.4363	0.9282	0.4616
P_{G5}	0.0631	0.1375	0.1347	0.1896	0.1308	0.1952
P_{G6}	0.4664	0.5364	0.5331	0.5988	0.5292	0.6131
Cost	**620.87**	649.24	**620.46**	657.59	**619.60**	651.71
Emission	0.2368	**0.2048**	0.2243	**0.2017**	0.2244	**0.2019**

Table 8 The best compromise solutions of NSGA

	***Case* 1**	***Case* 2**	***Case* 3**
P_{G1}	0.2252	0.2935	0.2712
P_{G2}	0.3622	0.3645	0.3670
P_{G3}	0.5222	0.5833	0.8099
P_{G4}	0.7660	0.6763	0.7550
P_{G5}	0.5397	0.5383	0.1357
P_{G6}	0.4187	0.4076	0.5239
Cost	**606.03**	**617.80**	**625.71**
Emission	**0.2041**	**0.2002**	**0.2136**

Table 9 The best compromise solutions of NPGA

	***Case* 1**	***Case* 2**	***Case* 3**
P_{G1}	0.2663	0.2976	0.2998
P_{G2}	0.3700	0.3956	0.4325
P_{G3}	0.5222	0.5673	0.7342
P_{G4}	0.7202	0.6928	0.6852
P_{G5}	0.5256	0.5201	0.1560
P_{G6}	0.4296	0.3904	0.5561
Cost	**608.90**	**617.79**	**630.06**
Emission	**0.2015**	**0.2004**	**0.2079**

***Case 3*:** MOEA techniques have been implemented and the trade-off fronts of different techniques for the best optimization runs are shown in Figure 9. In this case, the performance of NSGA is degraded with increasing the problem complexity. The best cost and best emission solutions obtained out of 10 runs are given in Table 7.

Table 10 The best compromise solutions of SPEA

	Case **1**	*Case* **2**	*Case* **3**
P_{G1}	0.2623	0.2752	0.3052
P_{G2}	0.3765	0.3752	0.4389
P_{G3}	0.5428	0.5796	0.7163
P_{G4}	0.6838	0.6770	0.6978
P_{G5}	0.5381	0.5283	0.1552
P_{G6}	0.4305	0.4282	0.5507
Cost	**610.30**	**617.57**	**629.59**
Emission	**0.2004**	**0.2001**	**0.2079**

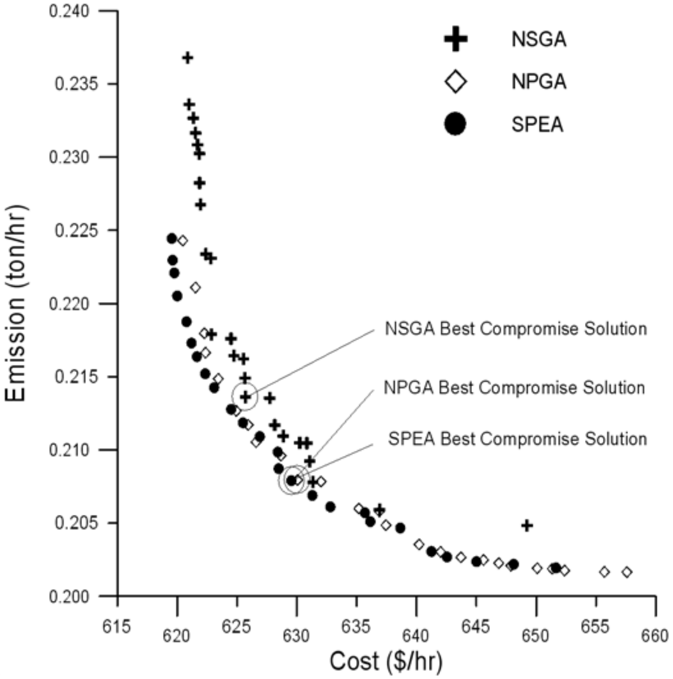


Fig. 9 Comparison of trade-off fronts, Case 3

Best compromise solution: - The membership functions given in Equation (27) and Equation (28) are used to evaluate each member of the non-dominated set for each technique. Then, the best compromise solution that has the maximum value of membership function was extracted. This procedure is applied in all cases and the best compromise solutions are given in Tables 8, 9, and 10 for NSGA, NPGA, and SPEA respectively. The best compromise solutions are also shown in Figures 8, 9, and 10. It is clear that there is good agreement between SPEA and NPGA.

7 A Comparative Study

Generally, the definition of quality in the case of multiobjective optimization is substantially more complex than for single objective optimization problems. This is because the optimization goal itself consists of the following multiple objectives [46, 48, 50]: -

1. The distance of the resulting non-dominated set to the Pareto-optimal front should be minimized.
2. A good distribution of the solutions found is desirable.
3. The spread of the obtained non-dominated solutions should be maximized.

In this section, the above results for the different techniques have been compiled and compared in view of the above objectives. In order to assess the diversity characteristics of the proposed techniques, the best fuel cost and the best emission solutions among the obtained non-dominated solutions for each technique given in Tables 4, 6, and 7 are compared to those of individual optimization of each objective given in Table 3. This indicates of how far the extreme solutions are from the single objective case. The agreement and closeness of the results given in these tables are quite evident as the best solutions of different techniques are almost identical. It can be concluded that the developed techniques have satisfactory diversity characteristics for the problem under consideration as the best solutions for individual optimization are obtained along with other non-dominated solutions in a single run.

A performance measure of the spread of the non-dominated solutions is presented in [46]. The measure estimates the range to which the fronts spread out. In other words, it measures the normalized distance of the two outer solutions, i.e. the best cost solution and the best emission solution. The average values of the normalized distance measure over 10 different optimization runs are given in Table 11. The results show that NPGA has the largest spread of the non-dominated solutions in Case 1 while SPEA has the largest spread in Case 2. In Case 3, NSGA has the largest spread.

On the other hand, the set coverage metric measure [50] for comparing the performance of different MOEAs has been examined in this study. The average values of this measure over 10 different optimization runs are given in Table 12. It can be shown that the non-dominated solutions of NSGA do not cover any SPEA solutions in Case 3 while those of NSGA are approximately covered by SPEA. In addition, NPGA non-dominated solutions barely cover SPEA solutions with a maximum coverage of 14.4% while SPEA solutions cover relatively higher percentages of NPGA solutions.

The quality measure [6] of the non-dominated solutions obtained by different MOEAs is applied. This quality measure starts with combining all individual non-dominated sets of all techniques to form a pool. An index to each solution is added to refer to the associated technique. Then, the dominance conditions are applied to all solutions in the pool. The non-dominated solutions are extracted from the pool to form an elite set of solutions obtained by all techniques. From their indices, the non-dominated solutions in the elite set can be classified according to their associated

Table 11 Normalized distance measure of different techniques

	NSGA	***NPGA***	***SPEA***
Case 1	0.93757	0.95001	0.93809
Case 2	0.92211	0.93747	0.94509
Case 3	0.85539	0.81312	0.85363

Table 12 Percentage of non-dominated solutions of set b covered by those in set a

Set A	***Set B***	***Case 1***	***Case 2***	***Case 3***
NSGA	***NPGA***	27.6	24.0	2.0
	SPEA	3.6	2.4	0.0
NPGA	***NSGA***	25.2	29.2	82.4
	SPEA	2.0	5.6	14.4
SPEA	***NSGA***	52.8	53.2	97.4
	NPGA	58.8	55.6	46.0

Table 13 Number of "Pareto-optimal" solutions of different techniques in elite set of non-dominated solutions

	NSGA	***NPGA***	***SPEA***	***Elite Set Size***
Case 1	36	16	129	181
Case 2	19	17	129	165
Case 3	1	35	81	117

Table 14 Normalized distance measure of different techniques on elite set of non-dominated solutions

	NSGA	***NPGA***	***SPEA***
Case 1	0.82937	0.73043	1.00000
Case 2	0.63184	0.93501	1.00000
Case 3	0.00000	0.53827	1.00000

technique. The quality measure has been applied to the non-dominated solutions obtained in each case. For 10 different optimization runs with 25 non-dominated solutions obtained by each technique per run, the created pool contains 750 solutions. For each case, the non-dominated solutions are extracted out of the pool and the elite set is formed. The elite set consists of 181, 165, and 117 for Cases 1, 2, and 3 respectively. The results of the proposed quality measure are given in Table 13. It can be observed that SPEA has the majority of the elite set members in all cases. It can be concluded that the non-dominated solutions obtained by SPEA are the best since approximately 71%, 78%, and 69% of the elite set size is contributed

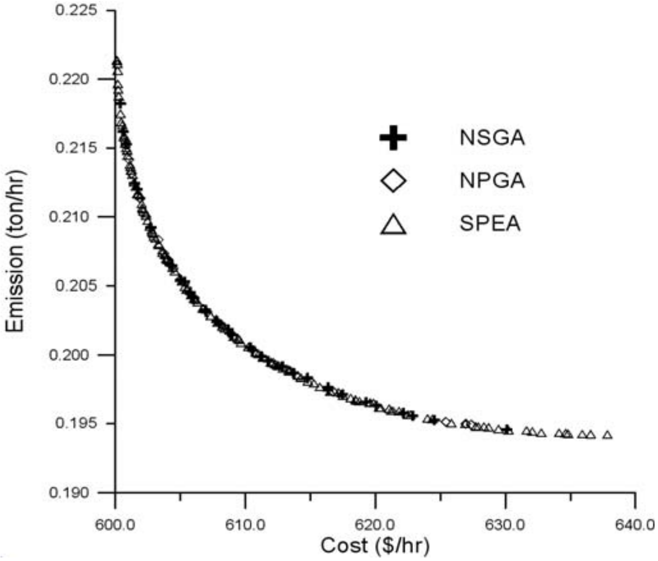


Fig. 10 The trade-off front of elite set of non-dominated solutions, Case 1

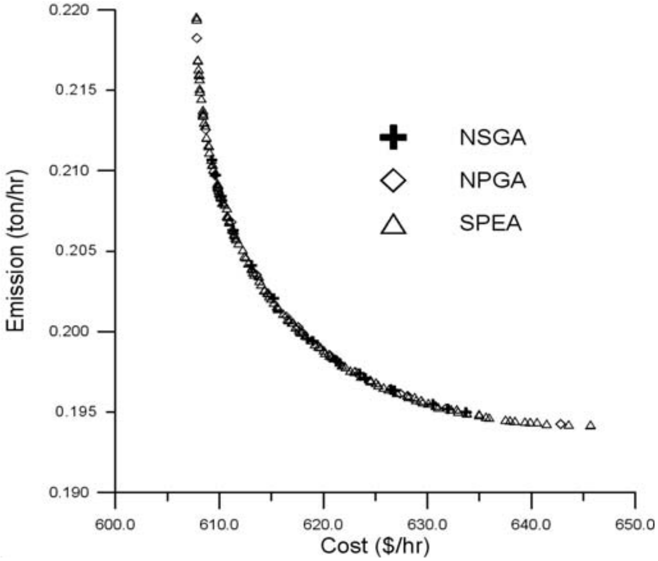


Fig. 11 The trade-off front of elite set of non-dominated solutions, Case 2

by SPEA in cases 1, 2, and 3 respectively. Also, it can be seen that only one non-dominated solution obtained by NSGA in case 3 is a member in the elite set. The trade-off represented by the non-dominated solutions in the elite set for all cases 1, 2, and 3 are shown in Figures 10, 11, and 12 respectively.

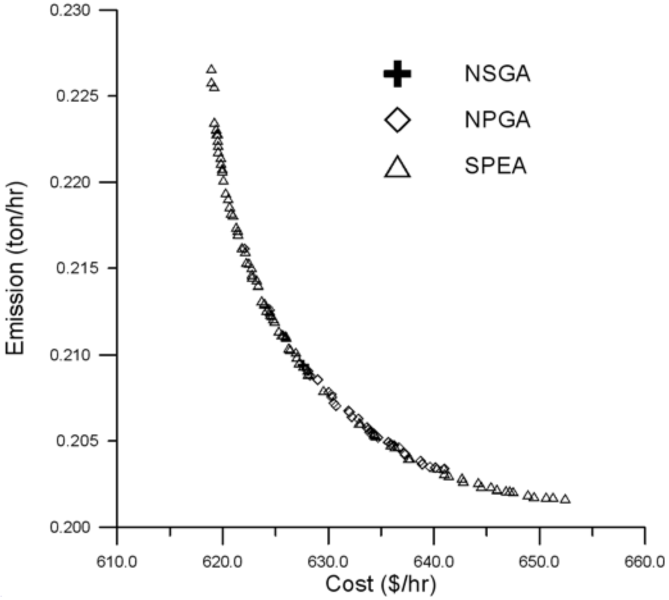


Fig. 12 The trade-off front of elite set of non-dominated solutions, Case 3

Table 15 Run time of Different Algorithms

	NSGA	***NPGA***	***SPEA***
Run time **(*s*)**	0.727	0.750	0.671

Table 16 Robustness of MOEA for different initial populations

	NSGA		***NPGA***		***SPEA***	
	Cost	*Emission*	*Cost*	*Emission*	*Cost*	*Emission*
Min	600.34	0.1946	600.31	0.1943	600.22	0.1942
Max	600.77	0.1949	600.78	0.1944	600.60	0.1943
Ave	600.43	0.1947	600.48	0.1943	600.33	0.1943

The average value of the normalized distance results of the proposed measure over 10 different optimization runs are given in Table 14. It is worth mentioning that the distance obtained with the proposed measure is that between the outer non-dominated solutions of each technique represented in the elite set. It can be seen that the non-dominated solutions obtained by SPEA span over the entire Pareto front in all cases. In general, it can be concluded that SPEA has the best distribution of the non-dominated solutions for the problem under consideration.

With the proposed approach of extracting an elite set from the combined non-dominated solutions of all techniques, it can be seen that the proposed measure and

Table 17 IEEE 30-bus test system line data

Line #	From Bus	To Bus	Resistance (pu)	Reactance (pu)	Susceptance (pu)	Rating (MVA)
1	1	2	0.0192	0.0575	0.0264	130
2	1	3	0.0452	0.1852	0.0204	130
3	2	4	0.0570	0.1737	0.0184	65
4	3	4	0.0132	0.0379	0.0042	130
5	2	5	0.0472	0.1983	0.0209	130
6	2	6	0.0581	0.1763	0.0187	65
7	4	6	0.0119	0.0414	0.0045	90
8	5	7	0.0460	0.1160	0.0102	70
9	6	7	0.0267	0.0820	0.0085	130
10	6	8	0.0120	0.0420	0.0045	32
11	6	9	0.0000	0.2080	0.0000	65
12	6	10	0.0000	0.5560	0.0000	32
13	9	11	0.0000	0.2080	0.0000	65
14	9	10	0.0000	0.1100	0.0000	65
15	4	12	0.0000	0.2560	0.0000	65
16	12	13	0.0000	0.1400	0.0000	65
17	12	14	0.1231	0.2559	0.0000	32
18	12	15	0.0662	0.1304	0.0000	32
19	12	16	0.0945	0.1987	0.0000	32
20	14	15	0.2210	0.1997	0.0000	16
21	16	17	0.0824	0.1923	0.0000	16
22	15	18	0.1070	0.2185	0.0000	16
23	18	19	0.0639	0.1292	0.0000	16
24	19	20	0.0340	0.0680	0.0000	32
25	10	20	0.0936	0.2090	0.0000	32
26	10	17	0.0324	0.0845	0.0000	32
27	10	21	0.0348	0.0749	0.0000	32
28	10	22	0.0727	0.1499	0.0000	32
29	21	22	0.0116	0.0236	0.0000	32
30	15	23	0.1000	0.2020	0.0000	16
31	22	24	0.1150	0.1790	0.0000	16
32	23	24	0.1320	0.2700	0.0000	16
33	24	25	0.1885	0.3292	0.0000	16
34	25	26	0.2544	0.3800	0.0000	16
35	25	27	0.1093	0.2087	0.0000	16
36	28	27	0.0000	0.3960	0.0000	65
37	27	29	0.2198	0.4153	0.0000	16
38	27	30	0.3202	0.6027	0.0000	16
39	29	30	0.2399	0.4533	0.0000	16
40	8	28	0.0636	0.2000	0.0214	32
41	6	28	0.0169	0.0599	0.0065	32

Table 18 IEEE 30-bus test system bus data

Bus	P_D (MW)	Q_D (MVAR)
1	0.00	0.00
2	21.70	12.70
3	2.40	1.20
4	7.60	1.60
5	94.20	19.00
6	0.00	0.00
7	22.80	10.90
8	30.00	30.00
9	0.00	0.00
10	5.80	2.00
11	0.00	0.00
12	11.20	7.50
13	0.00	0.00
14	6.20	1.60
15	8.20	2.50
16	3.50	1.80
17	9.00	5.80
18	3.20	0.90
19	9.50	3.40
20	2.20	0.70
21	17.50	11.20
22	0.00	0.00
23	3.20	1.60
24	8.70	6.70
25	0.00	0.00
26	3.50	2.30
27	0.00	0.00
28	0.00	0.00
29	2.40	0.90
30	10.60	1.90

the normalized distance measure are consistent and their results have a satisfactory agreement with the simulation results. Also, the proposed measure reflects properly the quality of the non-dominated solutions produced by each algorithm. In addition, several techniques can be compared in a single run rather than on a one-to-one basis.

The comparison of the average value of the run time over 10 different optimization runs per generation per "Pareto-optimal solution" of MOEA techniques with case 1 is given in Table 15. It is quite evident that the run time of SPEA is less than that of the other techniques.

The robustness of MOEA techniques with respect to different initial populations has been examined in all cases considered. Due to space limitations, only results for Case 1 are given in Table 16, which shows the minimum, the maximum, and the average values of the best cost and the best emission. It is clear that all techniques exhibit satisfactory degree of robustness to initial populations. In addition, SPEA gives better average results.

Based on the above comparisons and discussions, it can be concluded that SPEA is better than other techniques for the environmental/economic power dispatch optimization problem since elite solutions with satisfactory diversity characteristics have been produced in this study.

8 Future Work

Since this work represents an exploratory study aiming to demonstrate the potential of MOEA for solving EED problem, the fuel cost function given in Equation (1) is a smooth and simple quadratic one. However, more complicated formulations with non-smooth and non-convex fuel cost functions [30, 38, 51] can be considered in future work. Additionally, different objective functions, such as heat dispatch ,in addition to the fuel cost and emission objective functions [32] can be considered and incorporated in problem formulation in future studies.

On the other hand, new and revised versions of MOEA have been presented such as NSGA-II, NPGA 2, SPEA2, and multiobjective particle swarm optimization MOPSO. These techniques can be examined in future studies. This will enhance the potential of MOEA to solve more complex multiobjective power system optimization problems.

9 Conclusions

In this chapter, three multiobjective evolutionary algorithms have been compared and successfully applied to the environmental/economic power dispatch problem. The problem has been formulated as a multiobjective optimization problem with competing economic and environmental impact objectives. MOEAs have been compared to each other and to those reported in the literature. In addition, a new and efficient procedure for quality measure is proposed and compared to some measures reported in the literature. The optimization runs indicate MOEAs outperform the traditional techniques. Moreover, SPEA has better diversity characteristics and is more efficient when compared to other MOEAs. The results show that evolutionary algorithms are effective tools for handling multiobjective optimization where multiple trade-off solutions can be found in one simulation run.

In addition, the diversity of the non-dominated solutions is preserved. It is also demonstrated that SPEA has the best computational time. It can be concluded that MOEA has potential to solve different multiobjective power systems optimization problems.

Acknowledgements. The author acknowledges the support of King Fahd University of Petroleum & Minerals.

Appendix

The line and bus data of the IEEE 30-bus 6-generator system are given in Table 17 and Table 18 respectively.

References

1. Abido, M.A.: A New Multiobjective Evolutionary Algorithm for Environmental/Economic Power Dispatch. In: IEEE Power Engineering Society Summer Meeting, Vancouver, Canada, pp. 1263–1268 (2001)
2. Abido, M.A.: A Novel Multiobjective Evolutionary Algorithm for Solving Environmental/Economic Power Dispatch Problem. In: 14th Power Systems Computation Conference PSCC 2002, Session 41, Paper 2, Seville, Spain (2002)
3. Abido, M.A.: A Niched Pareto Genetic Algorithm for Multiobjective Environmental/Economic Dispatch. International Journal of Electrical Power and Energy Systems 25, 79–105 (2003)
4. Abido, M.A.: A novel multiobjective evolutionary algorithm for environmental/economic power dispatch. Electric Power Systems Research 65, 71–81 (2003)
5. Abido, M.A.: Environmental/Economic Power Dispatch Using Multiobjective Evolutionary Algorithms. IEEE Trans. on Power Systems 18, 1529–1537 (2003)
6. Abido, M.A.: Multiobjective Evolutionary Algorithms for Electric Power Dispatch Problem. IEEE Trans. on Evolutionary Computation 10, 315–329 (2006)
7. Abou El-Ela, A.A., Abido, M.A.: Optimal Operation Strategy for Reactive Power Control. Modelling, Simulation & Control, Part A 41, 19–40 (1992)
8. Brodesky, S.F., Hahn, R.W.: Assessing the Influence of Power Pools on Emission Constrained Economic Dispatch. IEEE Trans. on Power Systems 1, 57–62 (1986)
9. Xu, J., Chang, C., Wang, X.: Constrained Multiobjective Global Optimization of Longitudinal Interconnected Power System by Genetic Algorithm. IEE Proc.-Gener. Transm. Distrib. 143, 435–446 (1996)
10. Chang, C., Wong, K., Fan, B.: Security-Constrained Multiobjective Generation Dispatch Using Bicriterion Global Optimization. IEE Proc.-Gener. Transm. Distrib. 142, 406–414 (1995)
11. Coello, C.A.C., Christiansen, A.D.: MOSES: A Multiobjective Optimization Tool for Engineering Design. Engineering Optimization 31, 337–368 (1999)
12. Coello, C.A.C., Hernandez, F.S., Farrera, F.A.: Optimal Design of Reinforced Concrete Beams Using genetic Algorithms. Int. J. of Expert systems with Applications 12, 101–108 (1997)
13. Coello, C.A.C., Pulido, G.T., Lechuga, M.S.: Handling Multiple Objectives with Particle Swarm Optimization. IEEE Trans. on Evolutionary Computation 8, 256–279 (2004)
14. Das, D.B., Patvardhan, C.: New Multi-objective Stochastic Search Technique for Economic Load Dispatch. IEE Proc.-Gener. Transm. Distrib. 145, 747–752 (1998)
15. Deb, K., Pratab, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6, 182–197 (2002)
16. Dhillon, J.S., Parti, S.C., Kothari, D.P.: Stochastic Economic Emission Load Dispatch. Electric Power Systems Research 26, 179–186 (1993)
17. El-Keib, A.A., Ma, H., Hart, J.L.: Economic Dispatch in View of the Clean Air Act of 1990. IEEE Trans. On Power Systems 9, 972–978 (1994)

18. Farag, A., Al-Baiyat, S., Cheng, T.C.: Economic Load Dispatch Multiobjective optimization Procedures Using Linear Programming Techniques. IEEE Trans. on Power Systems 10, 731–738 (1995)
19. Farina, M., Amato, P.: A Fuzzy Definition of Optimality for Many Criteria Optimization Problems. IEEE Trans. On Systems, Man, and Cybernetics-Part A: Systems and Humans 34, 315–326 (2004)
20. Farina, M., Deb, K., Amato, P.: Dynamic Multiobjective Optimization Problems: Test Cases, Approximations, and Applications. IEEE Trans. on Evolutionary Computation 8, 425–442 (2004)
21. Fuller, R., Carlsson, C.: Fuzzy Multiple Criteria Decision Making: Recent Developments. Fuzzy Set and Systems 78, 139–153 (1996)
22. Fonseca, C.M., Fleming, P.J.: An Overview of Evolutionary Algorithms in Multiobjective Optimization. Evolutionary Computation 3, 1–16 (1995)
23. Granelli, G.P., Montagna, M., Pasini, G.L., Marannino, P.: Emission Constrained Dynamic Dispatch. Electric Power Systems Research 24, 56–64 (1992)
24. Helsin, J.S., Hobbs, B.F.: A Multiobjective Production Costing Model for Analyzing Emission Dispatching and Fuel Switching. IEEE Trans. on Power Systems 4, 836–842 (1989)
25. Herrera, F., Lozano, M., Verdegay, J.L.: Tackling Real-Coded genetic Algorithms: operators and Tools for Behavioral Analysis. Artificial Intelligence Review 12, 265–319 (1998)
26. Horn, J., Nafpliotis, N., Goldberg, D.E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In: Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, vol. 1, pp. 67–72. IEEE Service Center, Piscataway (1994)
27. Huang, C.M., Yang, H.T., Huang, C.L.: Bi-Objective Power Dispatch Using Fuzzy Satisfaction-Maximizing Decision Approach. IEEE Trans. on Power Systems 12, 1715–1721 (1997)
28. Kwang, Y.L., El-Sharkawi, M.A.: Modern Heuristic Optimization Techniques Theory and Applications to Power Systems. IEEE Press/ Wiley-Interscience, USA (2008)
29. Jensen, M.T.: Reducing the Run-Time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms. IEEE Trans. on Evolutionary Computation 7, 503–515 (2003)
30. Park, J.-B., Lee, K.-S., Shin, J.-R., Kwang, Y.L.: A Particle Swarm Optimization for Economic Dispatch With Nonsmooth Cost Functions. IEEE Trans. on Power Systems 20, 34–42 (2005)
31. Laumanns, M., Thiele, L., Zitzler, E.: Running Time Analysis of Multiobjective Evolutionary Algorithms on Pseudo-Boolean Functions. IEEE Trans. on Evolutionary Computation 8, 170–182 (2004)
32. Lingfeng, W., Chanan, S.: Stochastic Combined Heat and Power Dispatch Based on Multi-Objective Particle Swarm Optimization. In: Proceedings of 2006 IEEE Power Engineering Society General Meeting (2006)
33. Lis, J., Eiben, A.E.: A Multi-Sexual Genetic Algorithm for Multiobjective Optimization. In: Proceedings of the 1996 International Conference on Evolutionary Computation IEEE ICEC 1996, Nagoya, Japan, pp. 59–64 (1996)
34. Mahfoud, S.: Niching Methods for Genetic Algorithms, Ph. D. Thesis, Univ. of Illinois at Urbana-Champaign (1995)
35. Morse, J.N.: Reducing the Size of Nondominated Set: Pruning by Clustering. Computers and Operations Research 7, 55–66 (1980)
36. Sakawa, M., Yano, H., Yumine, T.: An Interactive Fuzzy Satisficing Method for Multiobjective Linear Programming Problems and Its Application. IEEE Trans. On Systems, Man, and Cybernetics 17, 654–661 (1987)

37. Schaffer, J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: Proceedings of the International Conference on Genetic Algorithms and Their Applications, Pittsburgh, July 24-26, 1985, pp. 93–100 (1985)
38. Selvakumar, A.I., Thanushkodi, K.: A New particle Swarm Optimization Solution to Nonconvex Economic Dispatch Problems. IEEE Trans. on Power Systems 22, 42–51 (2007)
39. Shin, S.Y., Lee, I.H., Kim, D., Zhang, B.T.: Multiobjective Evolutionary Optimization of DNA Sequences for Reliable DNA Computing. IEEE Trans. on Evolutionary Computation 9, 143–158 (2005)
40. Srinivas, N., Deb, K.: Multiobjective Function Optimization Using Nondominated Sorting Genetic Algorithms. Evolutionary Computation 2, 221–248 (1994)
41. Srinivasan, D., Chang, C.S., Liew, A.C.: Multiobjective Generation Schedule using Fuzzy Optimal Search Technique. IEE Proc.-Gener. Transm. Distrib. 141, 231–241 (1994)
42. Srinivasan, D., Tettamanzi, A.: An Evolutionary Algorithm for Evaluation of Emission Compliance Options in View of the Clean Air Act Amendments. IEEE Trans. on Power Systems 12, 152–158 (1997)
43. Talaq, J.H., El-Hawary, F., El-Hawary, M.E.: A Summary of Environmental/Economic Dispatch Algorithms. IEEE Trans. on Power Systems 9, 1508–1516 (1994)
44. Veldhuizen, D.A.V., Lamont, G.B.: Multiobjective Evolutionary algorithms: Analyzing the State-of-the-Art. Evolutionary Computation 8, 125–147 (2000)
45. Yokoyama, R., Bae, S.H., Morita, T., Sasaki, H.: Multiobjective Generation dispatch Based on Probability Security Criteria. IEEE Trans. on Power Systems 3, 317–324 (1988)
46. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8, 173–195 (2000)
47. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. TIK-Report, No. 103 (2001)
48. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms – A Comparative Case Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
49. Zitzler, E., Thiele, L.: An Evolutionary Algorithm for Multiobjective optimization: The Strength Pareto Approach. TIK-Report, No. 43 (1998)
50. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Trans. on Evolutionary Computation 3, 257–271 (1999)
51. Gaing, Z.-L.: Particle Swarm Optimization to Solving the Economic Dispatch Considering the Generator Constraints. IEEE Trans. on Power Systems 18, 1187–1195 (2003)

Fuzzy Evolutionary Algorithms and Genetic Fuzzy Systems: A Positive Collaboration between Evolutionary Algorithms and Fuzzy Systems

F. Herrera and M. Lozano

Abstract. There are two possible ways for integrating fuzzy logic and evolutionary algorithms. The first one involves the application of evolutionary algorithms for solving optimization and search problems related with fuzzy systems, obtaining genetic fuzzy systems. The second one concerns the use of fuzzy tools and fuzzy logic-based techniques for modelling different evolutionary algorithm components and adapting evolutionary algorithm control parameters, with the goal of improving performance. The evolutionary algorithms resulting from this integration are called fuzzy evolutionary algorithms. In this chapter, we shortly introduce genetic fuzzy systems and fuzzy evolutionary algorithms, giving a short state of the art, and sketch our vision of some hot current trends and prospects. In essence, we paint a complete picture of these two lines of research with the aim of showing the benefits derived from the synergy between evolutionary algorithms and fuzzy logic.

1 Introduction

Computational intelligence techniques such as artificial neural networks [157], fuzzy logic [204], and genetic algorithms (GAs) [87, 63] are popular research subjects, since they can deal with complex engineering problems which are difficult to solve by classical methods [109].

Hybrid approaches have attracted considerable attention in the computational intelligence community. One of the most popular approaches is the hybridization between fuzzy logic and GAs leading to genetic fuzzy systems (GFSs) [38] and fuzzy evolutionary algorithms [79, 149, 183]. Both are well known examples of a positive collaboration between soft computing techniques.

A GFS is basically a fuzzy rule based system (FRBS) augmented by a learning process based on evolutionary computation, which includes GAs, genetic programming, and evolution strategies, among other evolutionary algorithms (EAs) [56].

F. Herrera and M. Lozano
Department of Computer Science and Artificial Intelligence University of Granada,
18071 - Granada, Spain
e-mail: herrera@decsai.ugr.es, lozano@decsai.ugr.es

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 83–130.
springerlink.com

The automatic definition of a FRBS can be seen as an optimization or search problem, and GAs are a well known and widely used global search technique with the ability to explore a large search space for suitable solutions only requiring a performance measure. In addition to their ability to find near optimal solutions in complex search spaces, the generic code structure and independent performance features of GAs make them suitable candidates to incorporate a priori knowledge. In the case of FRBSs, this a priori knowledge may be in the form of linguistic variables, fuzzy membership function parameters, fuzzy rules, etc. These capabilities extended the use of GAs in the development of a wide range of approaches for designing FRBSs over the last few years.

The behaviour of the EAs in general, and GAs in particular, is strongly determined by the balance between exploration (to investigate new and unknown areas in a search space) and exploitation (to make use of knowledge acquired by exploration to reach better positions on the search space). The GA control parameter settings, such as mutation probability, crossover probability, and population size, are key factors in the determination of the exploitation versus exploration tradeoff. It has long been acknowledged that they have a significant impact on GA performance. If poor settings are used, the exploration/exploitation balance may not be reached in a profitable way; the GA performance shall be severely affected due to the possibility of premature convergence. Finding robust control parameter settings is not a trivial task, since their interaction with GA performance is a complex relationship and the optimal ones are problem-dependent. Furthermore, different control parameter values may be necessary during the course of a run to induce an optimal exploration/exploitation balance. For these reasons, adaptive GAs have been built that dynamically adjust selected control parameters or genetic operators during the course of evolving a problem solution. Their objective is to offer the most appropriate exploration and exploitation behaviour. FRBSs provide a tool which can convert the linguistic control strategy based on expert knowledge into an automatic control strategy. They are particularly suited to model the relationship between variables in environments that are either ill-defined or very complex. The adaptation of GA parameters is one such complex problem that may benefit from the use of FRBS, producing the so-called fuzzy adaptive GAs. If we consider any kind of EA that can be improved by means of fuzzy logic based techniques, then we can use the name of fuzzy EAs.

In this chapter we shortly introduce GFSs and fuzzy EAs, giving a short state of the art, and sketch our vision of some hot current trends and prospects.

The remainder of this article is organized as follows. In Section 2, we provide an overview of FRBSs. In Section 3, we focus our attention to GFSs. In Section 4, we tackle fuzzy EAs. Finally, in Section 5, we provide some concluding remarks of this work.

2 Fuzzy Rule Based Systems

FRBSs constitute one of the main contributions of fuzzy logic. The basic concepts which underlie these fuzzy systems are those of linguistic variables and fuzzy

IF-THEN rules. A linguistic variable, as its name suggests, is a variable whose values are words rather than numbers, e.g., small, young, very hot and quite slow. Fuzzy IF-THEN rules are of the general form: if antecedent(s) then consequent(s), where antecedent and consequent are fuzzy propositions that contain linguistic variables. A fuzzy IF-THEN rule is exemplified by “if the temperature is high then the fan-speed should be high”. With the objective of modelling complex and dynamic systems, FRBSs handle fuzzy rules by mimicking human reasoning (much of which is approximate rather than exact), reaching a high level of robustness with respect to variations in the system’s parameters, disturbances, etc. The set of fuzzy rules of an FRBS can be derived from subject matter experts or extracted from data through a rule induction process.

In this section, we present a brief overview of the foundations of FRBSs, with the aim of illustrating the way they behave. In particular, in Section 2.1, we introduce the important concepts of fuzzy sets and linguistic variables. In Section 2.2, we deal with the basic elements of FRBSs. Finally, in Section 2.3, we describe a simple instance of FRBS, a fuzzy logic controller for the inverted pendulum.

2.1 Preliminaries: Fuzzy Set and Linguistic Variable

A *fuzzy set* is distinct from a crisp set in that it allows its elements to have a degree of membership. The core of a fuzzy set is its membership function: a surface or line that defines the relationship between a value in the set’s domain and its degree of membership. In particular, according to the original ideal of Zadeh [208], membership of an element x to a fuzzy set A, denoted as $\mu_A(x)$ or simply $A(x)$, can vary from 0 (full non-membership) to 1 (full membership), i.e., it can assume all values in the interval $[0,1]$. Clearly, a fuzzy set is a generalization of the concept of a set whose membership function takes on only two values $\{0,1\}$.

The value of $A(x)$ describes a degree of membership of x in A. For example, consider the concept of *high temperature* in an environmental context with temperatures distributed in the interval [0, 50] defined in degree centigrade. Clearly 0^{o}C is not understood as a high temperature value, and we may assign a null value to express its degree of compatibility with the high temperature concept. In other words, the membership degree of 0^{o}C in the class of high temperatures is zero. Likewise, 30^{o}C and over are certainly high temperatures, and we may assign a value of 1 to express a full degree of compatibility with the concept. Therefore, temperature values in the range [30, 50] have a membership value of 1 in the class of high temperatures. From 0^{o}C to 30^{o}C, the degree of membership in the fuzzy set high temperature gradually increases, as exemplified in Figure 1, which actually is a membership function $A : T \rightarrow [0,1]$ characterizing the fuzzy set of high temperatures in the universe $T = [0,50]$. In this case, as temperature values increase they become more and more compatible with the idea of high temperature.

Linguistic variables are variables whose values are not numbers but words or sentences in a natural or artificial language. This concept has clearly been developed as a counterpart to the concept of a numerical variable. More precisely, a linguistic

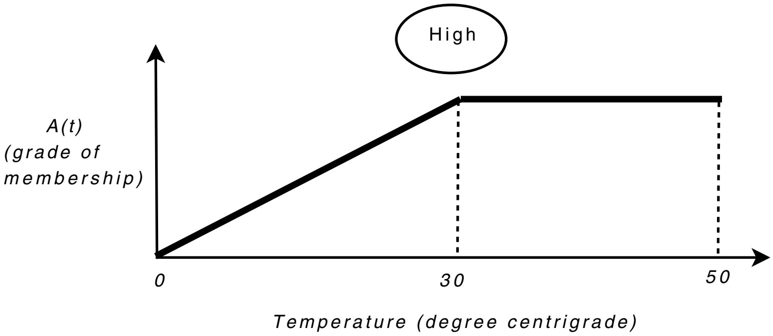


Fig. 1 Membership function

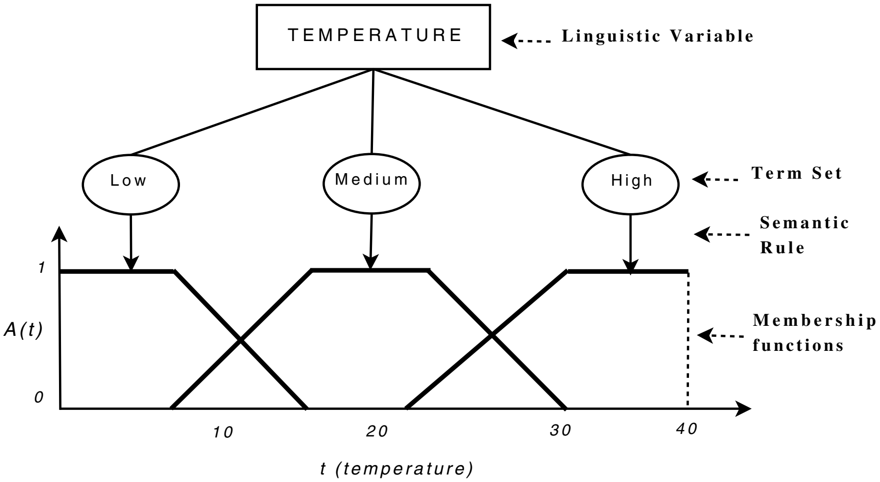


Fig. 2 Example of linguistic variable *Temperature* with three linguistic terms

variable L is defined as a quintuple [107]: $L = (x, A, X, g, m)$, where x is the base variable, $A = \{A_1, A_2, \ldots, A_N\}$ is the set of *linguistic terms* of L (called *term-set*), X is the domain (universe of discourse) of the base variable, g is a syntactic rule for generating linguistic terms and m is a semantic rule that assigns to each linguistic term its *meaning* (a fuzzy set in X). Figure 2 shows an example of a linguistic variable *Temperature* with three linguistic terms "Low, Medium, and High". The base variable is the temperature given in appropriate physical units.

Each underlying fuzzy set defines a portion of the variable's domain; but this portion is not uniquely defined. Fuzzy sets overlap as a natural consequence of their elastic boundaries. Such an overlap not only implements a realistic and functional semantic mechanism for defining the nature of a variable when it assumes various data values but provides a smooth and coherent transition from one state to another.

2.2 Basic Elements of FRBSs

The essential part of FRBSs is a set of IF-THEN linguistic rules, whose antecedents and consequents are composed of fuzzy statements, related by the dual concepts of fuzzy implication and the compositional rule of inference.

An FRBS is composed of a *knowledge base* (KB), that includes the information in the form of IF-THEN fuzzy rules;

IF *a set of conditions are satisfied*
THEN *a set of consequents can be inferred*

and an *inference engine module* that includes:

- A *fuzzification interface*, which has the effect of transforming crisp data into fuzzy sets.
- An *inference system*, that uses them together with the KB to make inference by means of a reasoning method.
- A *defuzzification interface*, that translates the fuzzy rule action thus obtained to a real action using a defuzzification method.

FRBSs can be broadly categorized into different families:

- The first includes linguistic models based on collections of IF-THEN rules, whose antecedents are linguistic values, and the system behaviour can be described in natural terms. The consequent is an output action or class to be applied. For example, we can denote them as:
  R_i : If X_{i1} is A_{i1} and $\cdots$ and X_{in} is A_{in} then Y is B_i
  or
  R_i : If X_{i1} is A_{i1} and $\cdots$ and X_{in} is A_{in} then C_k with w_{ik}
  with $i = 1$ to M, and with X_{i1} to X_{in} and Y being the input and output variables for regression respectively, and C_k the output class associated to the rule for classification, with A_{i1} to A_{in} and B_i being the involved antecedents and consequent labels, respectively, and w_{ik} the certain factor associated to the class. They are usually called *linguistic* FRBSs or *Mamdani* FRBSs [134].
- The second category is based on a rule structure that has fuzzy antecedent and functional consequent parts. This can be viewed as the expansion of piece-wise linear partition represented as
  R_i : If X_{i1} is A_{i1} and $\cdots$ and X_{in} is A_{in} then $Y = p(X_{i1}, \cdots, X_{in})$,
  with $p(\cdot)$ being a polynomial function, usually a linear expression, $Y = p_0 + p_1 \cdot X_{i1} + \cdots + p_n \cdot X_{in}$. The approach approximates a nonlinear system with a combination of several linear systems. They are called *Takagi and Sugeno's* type fuzzy systems [177] (TS-type fuzzy systems).
- Other kinds of fuzzy models are the approximate or scatter partition FRBSs, which differ from the linguistic ones in the direct use of fuzzy variables [4]. Each fuzzy rule thus presents its own semantic, i.e., the variables take different fuzzy sets as values (and not linguistic terms from a global term set). The fuzzy rule structure is then as follow:

R_i : If X_{i1} is $\hat{A}_{i1}$ and $\cdots$ and X_{in} is $\hat{A}_{in}$ then Y is $\hat{G}_i$
with $\hat{A}_{ij}$ to $\hat{A}_{in}$ and $\hat{G}_i$ being fuzzy sets. The major difference with respect to the rule structure considered in linguistic FRBSs is that rules of an approximate nature are semantics free, whereas descriptive rules operate in the context formulated by means of the linguistic semantics.

In linguistic FRBSs, the KB is composed of two components, a *data base* (DB) and a *rule base* (RB).

- A DB, containing the linguistic term sets considered in the linguistic rules and the membership functions defining the semantics of the linguistic labels.
  Each linguistic variable involved in the problem will have an associated fuzzy partition of its domain representing the fuzzy set associated with each of its linguistic terms. Figure 5 shows an example of a fuzzy partition with five labels. This can be considered as a discretization approach for continuous domains where we establish a membership degree to the items (labels), we have an overlapping between them, and the inference engine manages the matching between the patterns and the rules, providing an output according to the rule consequents with a positive matching. The determination of the fuzzy partitions is crucial in fuzzy modelling [11], and the granularity of the fuzzy partition plays an important role for the FRBS behaviour [39].

  If we manage approximate FRBSs, then we do not have a DB due to the fact that rules have associated the fuzzy values.
- An RB, comprises a collection of linguistic rules that are joined by a rule connective (”also” operator). In other words, multiple rules can fire simultaneously for the same input.

The inference engine of FRBSs acts in a different way depending on the kind of problem (classification or regression) and the kind of fuzzy rules (linguistic ones, TS-ones, etc). It always includes a fuzzification interface that serves as the input to the fuzzy reasoning process, an inference system that infers from the input to several resulting output (fuzzy set, class, etc) and the defuzzification interface or output interface that converts the fuzzy sets obtained from the inference process into a crisp action that constitutes the global output of the FRBS, in the case of regression problems, or provide the final class associated to the input pattern according to the inference model.

The generic structure of an FRBS is shown in Figure 3.

For more information about fuzzy systems the following books may be consulted [204, 113, 38, 94]. For different issues associated with the trade-off between the interpretability and accuracy of FRBSs, the two following edited books present a collection of contributions on the topic [25, 26].

Finally, we must point out that we can find a lot of applications of FRBSs in all areas of engineering, sciences, medicine, etc. At present it is very easy to search for these applications using the publisher web search tools focusing the search in journals of different application areas.

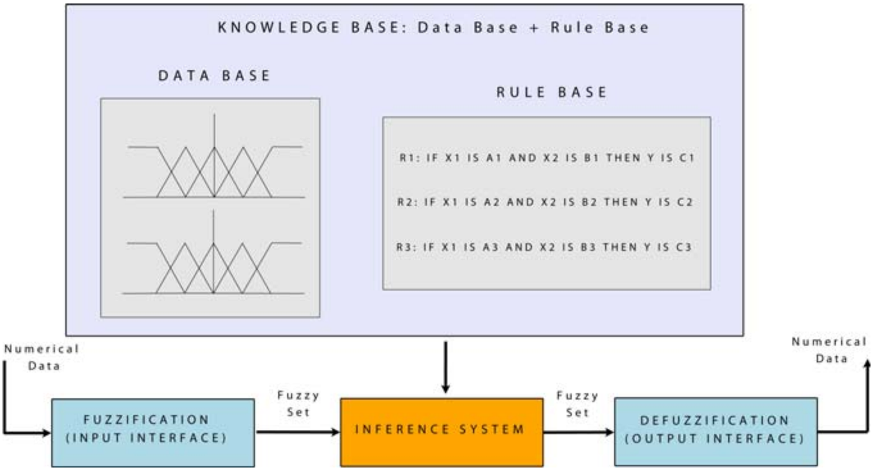

KNOWLEDGE BASE: Data Base + Rule Base
DATA BASE
RULE BASE
R1: IF X1 IS A1 AND X2 IS B1 THEN Y IS C1
R2: IF X1 IS A2 AND X2 IS B2 THEN Y IS C2
R3: IF X1 IS A3 AND X2 IS B3 THEN Y IS C3
Numerical Data
FUZZIFICATION (INPUT INTERFACE)
Fuzzy Set
INFERENCE SYSTEM
Fuzzy Set
DEFUZZIFICATION (OUTPUT INTERFACE)
Numerical Data

Fig. 3 Structure of an FRBS

2.3 Example of FRBS: Fuzzy Logic Control of an Inverted Pendulum

Fuzzy logic controllers [53] are a particular model of FRBS that provide a tool which can convert the linguistic control strategy based on expert knowledge into an automatic control strategy. In these controllers, the domain knowledge is represented by a set of fuzzy IF-THEN rules that approximate a mapping from a state space **X** to an output space **Y**. They have been used in many practical applications, especially industrial ones in Japan and Europe. Industrial success stories of fuzzy control include portable video cameras, automatic transmission of automobiles, furnace temperature, robotics, urban underground railway, and banking.

The example of the inverted pendulum given in [205] is selected to illustrate elementary fuzzy control principles. Consider the problem of keeping an inverted pendulum (which is fixed) articulated at a fixed point on a mobile cart. The cart can move forward and backward, and the controller decides on the direction and acceleration of the cart (Figure 4).

To balance an upright pendulum, we know from naive physics that the control force F should be chosen according to the magnitudes of the input variables θ and ω that measure the angle from the upright position and the angular velocity, respectively. The relation between these variables is linguistic, a much weaker form than differential equations. That is exactly what happens in a human mind that processes information qualitatively. Humans choose F using common sense knowledge in the form of rules such as “if the pendulum is in a balanced position, then hold very still, that is, do not apply any force”. By taking all such rules into account, the inverted pendulum can be successfully controlled.

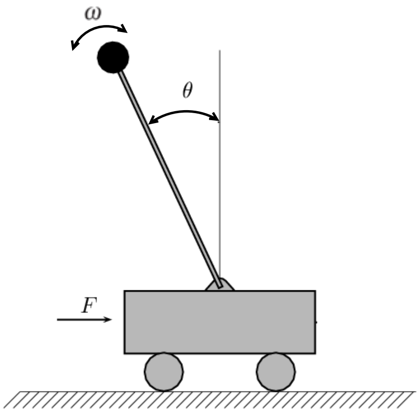


Fig. 4 Inverted pendulum

A sensor measures θ and ω (state variables) and a fuzzy logic controller may adjust F (output or control space) via a real time feedback loop with the objective of taking the pendulum to the vertical position. While the classical equations of motion of this system are extremely complicated and depend upon the specific characteristics of the pendulum (mass distribution, length), Yamakawa [205] found a set of linguistic fuzzy rules providing a stable fuzzy control of the pendulum independently of its characteristics. They are the following:

Rule 1. IF θ is PM AND ω is ZR THEN F is PM.
Rule 2. IF θ is PS AND ω is PS THEN F is PS.
Rule 3. IF θ is PS AND ω is NS THEN F is ZR.
Rule 4. IF θ is NM AND ω is ZR THEN F is NM.
Rule 5. IF θ is NS AND ω is NS THEN F is NS.
Rule 6. IF θ is NS AND ω is PS THEN F is ZR.
Rule 7. IF θ is ZR AND ω is ZR THEN F is ZR.

The linguistic term set for θ, ω, and F is $\{$Negative Large (NL), Negative Medium (NM), Negative Small (NS), Zero (ZR), Positive Small (PS), Positive Medium (PM), Positive Large (PL)$\}$, which has associated the fuzzy partition of their corresponding domains shown in Figure 5.

Given a sensor measured state (θ, ω), the inference obtained from the fuzzy controller is the result of interpolating among the response of these linguistic fuzzy rules. The inference's outcome is a membership function defined on the output space, which is then aggregated (defuzzified) to produce a crisp output.

The fuzzy logic controller described above is an example of linguistic FRBS. However, the problem of controlling the inverted pendulum may be tackled as well by means of a fuzzy logic controller based on the TS-type fuzzy system model. In this case, possible TS-type rules may include:

If θ is ZR and ω is ZR then $F = 0$.
If θ is PS and ω is ZR then $F = 0.5 \times \theta$.
If θ is PS and ω is NS then $F = 0.4 \times \theta + 0.6 \times \omega$.

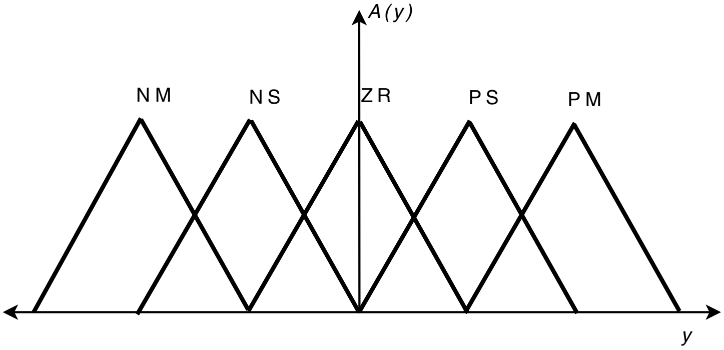


Fig. 5 Membership functions of the linguistic variables (where y stands for θ, ω, and F)

3 Genetic Fuzzy Systems

FRBSs constitute an extension to classical rule-based systems, because they deal with ”IF-THEN” rules, whose antecedents and consequents are composed of fuzzy logic statements, instead of classical ones. They have demonstrated their ability for control problems [146], modelling [148], classification or data mining [113, 94] in a huge number of applications.

A GFS is basically a fuzzy system augmented by a learning process based on evolutionary computation, which includes GAs, genetic programming, and evolution strategies, among other EAs. Figure 6 illustrates this idea, where the genetic process learns or tunes different components of an FRBS.

The central aspect of the use of a GA for automatic learning of an FRBS is that the KB design process can be analyzed as an optimization problem.

From the optimization point of view, to find an appropriate KB is equivalent to coding it as a parameter structure and then finding the parameter values that give us the optimum for a fitness function. The KB parameters provide the search space that is transformed according to a genetic representation. Therefore, the first step in designing a GFS is to decide which parts of the KB are subject to optimization by the GA.

In the last few years we observe an increase of published papers in the topic due to the high potential of GFSs. In contrast to neural networks, clustering, rule induction and many other machine learning approaches, GAs provide a means to encode and evolve rule antecedent aggregation operators, different rule semantics, rule base aggregation operators and defuzzification methods. Therefore, GAs remain today as one of the few knowledge acquisition schemes available to design and, in some sense, optimize FRBSs with respect to the design decisions, allowing decision makers to decide what components are fixed and which ones evolve according to the performance measures.

The predominant type of GFS is that focused on FRBSs. However other kinds of GFSs have been developed, with successful results. They include genetic fuzzy

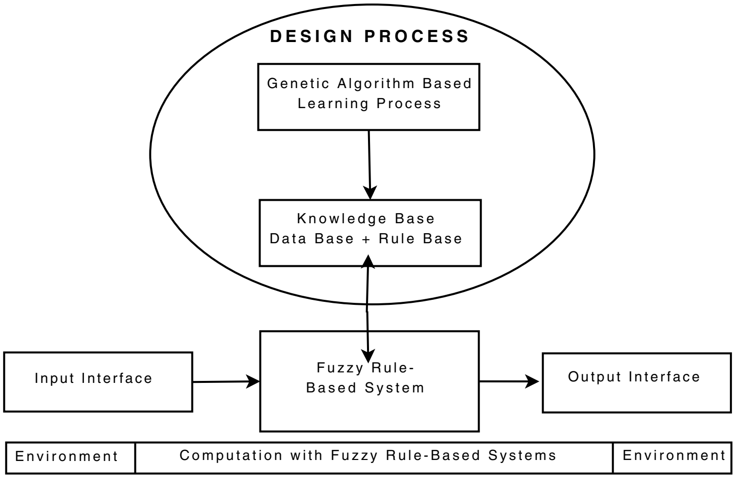

DESIGN PROCESS
Genetic Algorithm Based Learning Process
Knowledge Base Data Base + Rule Base
Input Interface
Fuzzy Rule-Based System
Output Interface
Environment
Computation with Fuzzy Rule-Based Systems
Environment

Fig. 6 Genetic fuzzy systems

neural networks and genetic fuzzy clustering algorithms. We will not analyze them in this papers. Readers can find an extended introduction to them in [38] (chapter 10).

In this section, we propose a taxonomy of GFSs focused on the FRBS components and sketch our vision of some hot current trends of GFSs [73].

3.1 Taxonomy of Genetic Fuzzy Systems

The central aspect on the use of GAs for automatic learning of FRBSs is that the design process can be analyzed as a search problem in the space of models, such as the space of rule sets, by means of the coding of the model in a chromosome.

From the optimization point of view, to find an appropriate fuzzy model is equivalent to code it as a parameter structure and then to find the parameter values that give us the optimum for a concrete fitness function. Therefore, the first step in designing a GFS is to decide which parts of the fuzzy system are subjected to optimization by the GA coding them into chromosomes.

We divide the GFS approaches into two processes, tuning and learning. It is difficult to make a clear distinction between tuning and learning processes, since establishing a precise borderline becomes as difficult as defining the concept of learning itself. The first fact that we have to take into consideration is the existence or not of a previous KB, including DB and RB. In the framework of GFSs we can briefly introduce the following distinction.

- Genetic tuning. If there exists a KB, we apply a genetic tuning process for improving the FRBS performance but without changing the existing RB. That is, to adjust FRBS parameters for improving its performance, maintaining the same RB.

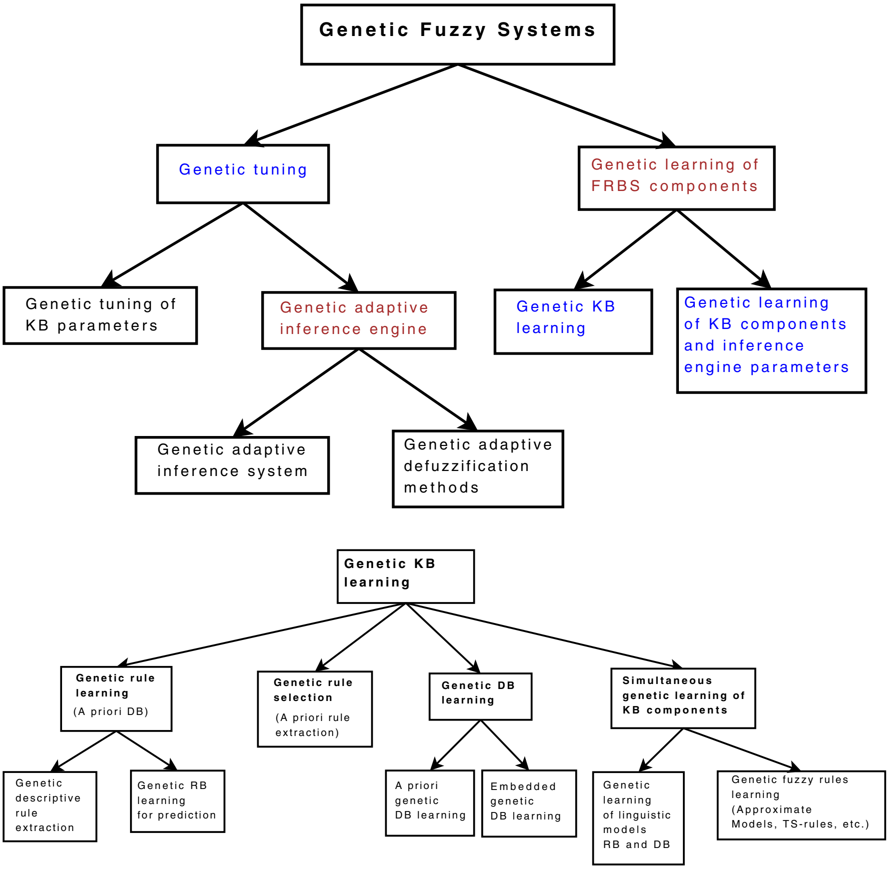

Genetic Fuzzy Systems
Genetic tuning
Genetic learning of FRBS components
Genetic tuning of KB parameters
Genetic adaptive inference engine
Genetic KB learning
Genetic learning of KB components and inference engine parameters
Genetic adaptive inference system
Genetic adaptive defuzzification methods
Genetic KB learning
Genetic rule learning (A priori DB)
Genetic rule selection (A priori rule extraction)
Genetic DB learning
Simultaneous genetic learning of KB components
Genetic descriptive rule extraction
Genetic RB learning for prediction
A priori genetic DB learning
Embedded genetic DB learning
Genetic learning of linguistic models RB and DB
Genetic fuzzy rules learning (Approximate Models, TS-rules, etc.)

Fig. 7 GFSs Taxonomy

- Genetic learning. The second possibility is to learn KB components (where we can even include an adaptive inference engine). That is, to involve the learning of KB components among other FRBS components.

We classify the proposals according to these two processes and according to the FRBS components involved in the genetic learning process. In this way, we consider the taxonomy shown in Figure 7 [73].

There are three main areas in the taxonomy that we can observe in the first tree: genetic tuning, genetic KB learning, and genetic learning of KB components and inference engine parameters.

In the following, we briefly analyze the three areas. We will provide some references as examples for every approach, but we do not present an exhaustive list of papers, this is far from the chapter's objective.

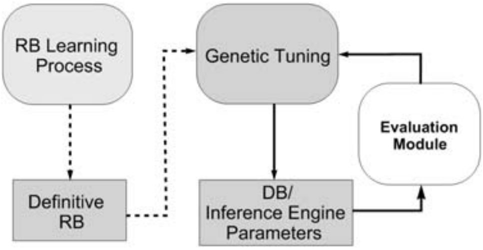


Fig. 8 Genetic tuning process

Genetic tuning

With the aim of making the FRBS perform better, some approaches try to improve the preliminary DB definition or the inference engine parameters once the RB has been derived. A graphical representation of this kind of tuning is shown in Figure 8.

The following three tuning possibilities can be considered (see the sub-tree under "genetic tuning").

1. *Genetic tuning of KB parameters.* In order to do so, a tuning process considering the whole KB obtained (the preliminary DB and the derived RB) is used a posteriori to adjust the membership function parameters. Nevertheless, the tuning process only adjusts the shapes of the membership functions and not the number of linguistic terms in each fuzzy partition, which remains fixed from the beginning of the design process. In [100], we can find a first and classic proposal on tuning. We can also find recent proposals that introduce linguistic modifiers for tuning the membership functions, see [24]. This latter approach is close to the inference engine adaptation.
2. *Genetic adaptive inference systems.* The main aim of this approach is the use of parameterized expressions in the Inference System, sometimes called Adaptive Inference Systems, for getting higher cooperation among the fuzzy rules and therefore more accurate fuzzy models without loosing the linguistic rule interpretability. In [8, 42, 43], we can find proposals in this area focused in regression and classification.
3. *Genetic adaptive defuzzification methods.* The most popular technique in practice, due to its good performance, efficiency and easier implementation, is to apply the defuzzification function to every inferred rule fuzzy set (getting a characteristic value) and to compute them by a weighted average operator. This method introduces the possibility of using parameter based average functions, and the use of GAs can allow us to adapt the defuzzification methods. In [105], we can find a proposal in this area.

Genetic KB learning

As a second big area we find the learning of KB components. We will now describe the four approaches that can be found within the genetic learning of a KB (see the second tree under "genetic KB learning").

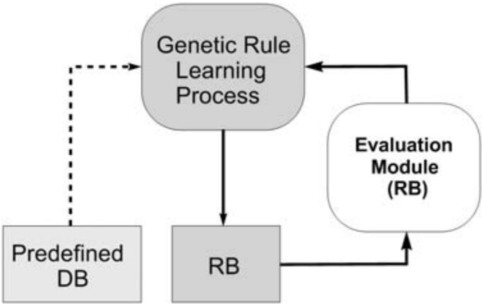


Fig. 9 Genetic rule learning process

1. *Genetic rule learning.* Most of the approaches proposed to automatically learn the KB from numerical information have focused on the RB learning, using a predefined DB. The usual way to define this DB involves choosing a number of linguistic terms for each linguistic variable (an odd number between 3 and 9, which is usually the same for all the variables) and setting the values of the system parameters by an uniform distribution of the linguistic terms into the variable universe of discourse. Figure 9 shows this type of RB learning graphically. The pioneer proposal for this approach can be found in [180].

   On the other hand, we also find approaches that are focused on the extraction of some descriptive rules for data mining problems (association rules, subgroup discovery, etc.) [102, 48].
2. *Genetic rule selection.* Sometimes we have a large number of rules extracted via a data mining method that subsequently provide us with a large number of rules associated with our problem. A big RB and an excessive number of rules makes it difficult to understand the FRBS behaviour. Thus we can find different kinds of rules in a fuzzy rule set: irrelevant rules, redundant rules, erroneous rules and conflictive rules, which perturb the FRBS performance when they coexist with others. To face this problem we can use a genetic rule selection process for obtaining an optimized subset of rules from a previous fuzzy rule set, by selecting some of them. Figure 10 illustrates this idea graphically. In [95] we can find the most classic and first contribution in this area and in [92] we can find the first journal paper on multiobjective genetic rule selection.

   We must point out that rule selection can be combined with tuning approaches, to try to get a good rule set together with a tuned set of parameters. In [24, 5], we can find two recent proposal that combines genetic tuning with rule selection.
3. *Genetic DB learning.* There is another way to generate the whole KB that considers two different processes to derive each component, DB and RB. A DB generation process allows us to learn the shape or the membership functions and other DB components such as the scaling functions, the granularity of the fuzzy partitions, etc. This DB generation process can use a measure for evaluating the quality of the DB, we can call it “a priori genetic DB learning”. The second possibility is to consider an embedded genetic learning process where the DB generation process wraps an RB learning one working as follows: each time a

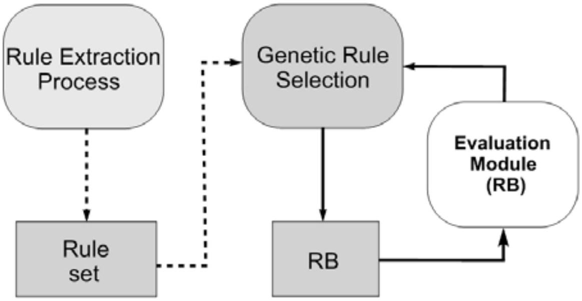


Fig. 10 Genetic rule selection process

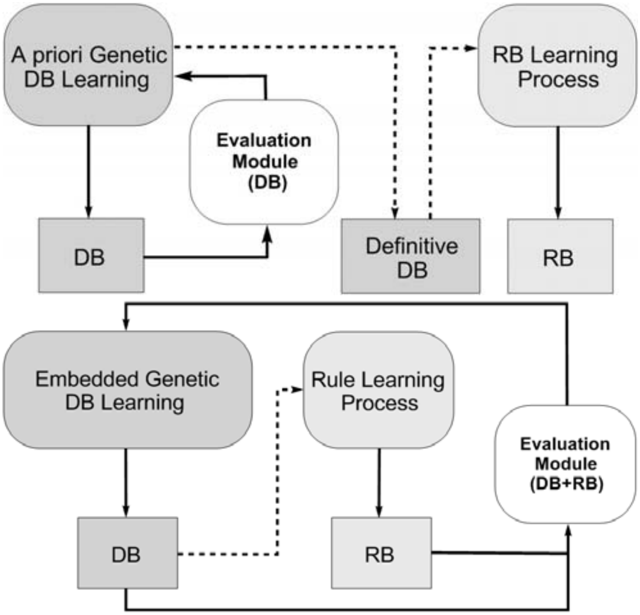


Fig. 11 Genetic DB learning (embedded and a priori)

DB has been obtained by the DB definition process, the RB generation method is used to derive the rules, and some type of error measure is used to validate the whole KB obtained. We should note this operation mode involves a partitioning of the KB learning problem. These two kinds of learning models are represented in Figure 11. In [41], we can find a proposal following the embedded genetic DB learning.

4. *Simultaneous genetic learning of KB components.* Other approaches try to learn the two components of the KB simultaneously. This kind of learning is depicted in Figure 12. Working in this way, they have the possibility of generating better definitions but there is a need to deal with a larger search space that makes the

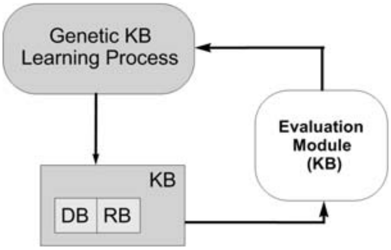


Fig. 12 Genetic KB learning process

learning process more difficult and slow. In [85], we can find a contribution that uses the simultaneous genetic KB learning process.

Genetic learning of KB components and inference engine parameters

This is the last area of GFSs taxonomy, belonging to a hybrid model between an adaptive inference engine and KB components learning. We can find novel approaches that try to find high cooperation between the inference engine via parameter adaptation and the learning of KB components, including both in a simultaneous learning process. In [135], we can find a recent proposal to learn a linguistic RB and the parametric aggregation connectors of the inference and defuzzification in a single step. Figure 13 presents the coding scheme of the model proposed in this paper.

3.2 *Genetic Learning: Rule Coding and Cooperation/Competition Evolutionary Process*

Although GAs were not specifically designed for learning, but rather as global search algorithms, they offer a set of advantages for machine learning. Many methodologies for machine learning are based on the search for a good model inside the space of possible models. In this sense, they are very flexible because the same GA can be used with different representations. Genetic learning processes cover

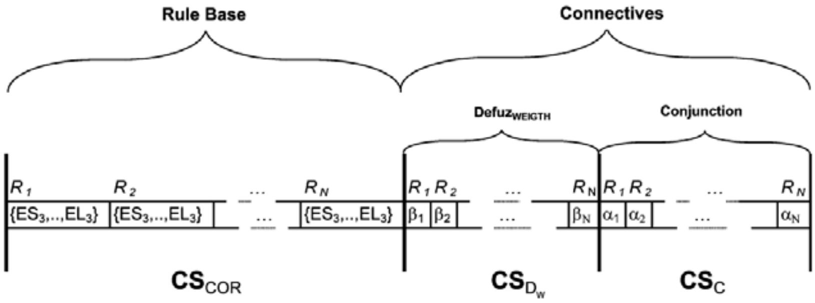


Fig. 13 Example of the coding scheme for learning an RB and the inference connective parameters

different levels of complexity according to the structural changes produced by the algorithm, from the simplest case of parameter optimization to the highest level of complexity for learning the rule set of a rule-based system, via the coding approach and the cooperation or competition between chromosomes.

When considering the task of learning rules in a rule based system, a wider range of possibilities is open. When considering a rule based system and focusing on learning rules, the different genetic learning methods follow two approaches in order to encode rules within a population of individuals:

- The "Chromosome = Set of rules", also called the Pittsburgh approach, in which each individual represents a rule set (Smith 1980). In this case, a chromosome evolves a complete RB and they compete among them along the evolutionary process. GABIL is a proposal that follows this approach [47].
- The "Chromosome = Rule" approach, in which each individual codifies a single rule, and the whole rule set is provided by combining several individuals in a population (rule cooperation) or via different evolutionary runs (rule competition). In turn, within the "Chromosome = Rule" approach, there are three generic proposals:
  - The Michigan approach, in which each individual encodes a single rule. These kinds of systems are usually called learning classifier systems [88]. They are rule-based, message-passing systems that employ reinforcement learning and a GA to learn rules that guide their performance in a given environment. The GA is used for detecting new rules that replace the bad ones via the competition between the chromosomes in the evolutionary process. An interesting study on the topic can be found in [110].
  - The IRL (Iterative Rule Learning) approach, in which each chromosome represents a rule. Chromosomes compete in every GA run, choosing the best rule per run. The global solution is formed by the best rules obtained when the algorithm is run multiple times. SIA [188] is a proposal that follows this approach.
  - The GCCL (genetic cooperative-competitive learning) approach, in which the complete population or a subset of it encodes the RB. In this model the chromosomes compete and cooperate simultaneously. COGIN [67], REGAL [62] and LOGENPRO [200] are examples with this kind of representation.

These four genetic learning approaches (Pittsburgh, Michigan, IRL and GCCL) have been considered for learning KB components, and we can find different examples of them in the literature. Two of the pioneer GFS proposals were focused on the Pittsburgh [180] and Michigan [186] approaches. MOGUL [37, 83, 35] and SLAVE [64] are two proposals that follow the IRL approach in the framework of GFSs. In [93, 97], we find two proposals following the GCCL approach.

3.3 *Some GFS Milestones: Books and Special Issues*

For beginners, in the following we present the GFS milestones associated to the books and special issues published in the specialized literature.

We can find two authored books and three edited ones:

- A, Geyer-Schulz. Fuzzy Rule-Based Expert Systems and Genetic Machine Learning. Physica-Verlag, 1995. This is the first GFS book. It is a very specific book focused on fuzzy classifier systems (Michigan approach) and RB learning with genetic programming.
- O. Cordón, F. Herrera, F. Hoffmann and L. Magdalena. Genetic Fuzzy Systems. Evolutionary Tuning and Learning of Fuzzy Knowledge Bases, World Scientific, 2001. This is the first general GFS book. It covers the overall state of the art of GFSs at that time.

These three following books compile an important number of contributions that gave maturity to the topic.

- F. Herrera and J.L. Verdegay (Eds.). Genetic Algorithms and Soft Computing. Physica-Verlag, 1996.
- E. Sanchez, Shibata and L. Zadeh (Eds.). Genetic Algorithms and Fuzzy Logic Systems. Soft Computing Perspectives. World Scientific, 1997.
- W. Pedrycz (Ed.). Fuzzy Evolutionary Computation. Kluwer Academic Publishers, 1997.

In the following we provide a list of the journal special issues devoted to GFSs, including important contributions to all topics of GFSs.

- F. Herrera. Special Issue on Genetic Fuzzy Systems for Control and Robotics. International Journal of Approximate Reasoning, Volume 17, Number 4, November 1997.
- F. Herrera and L. Magdalena. Special Issue on Genetic Fuzzy Systems. International Journal of Intelligent Systems, Volume 13, Numbers 10-11, Oct.-Nov. 1998.
- O. Cordón, F. Herrera, F. Hoffmann and L. Magdalena. Special Issue on Recent Advances in Genetic Fuzzy System. Information Sciences, Volume 136, Numbers 1-4 , August 2001.
- O. Cordón, F. Gomide, F. Herrera, F. Hoffmann, L. Magdalena. Special Issue on Genetic Fuzzy Systems. Fuzzy Sets and Systems, Volume 141, Number 1, January 2004.
- J. Casillas, M.J. del Jesus, F. Herrera, R. Pérez, P. Villar. Special Issue on Genetic Fuzzy Systems and the Interpretability-Accuracy Trade-off. International Journal of Approximate Reasoning. Volume 44, Number 1, February 2007.
- O. Cordón, R. Alcalá, J. Alcalá-Fdez, I. Rojas. Genetic Fuzzy Systems. Special Section on Genetic Fuzzy Systems: What's Next?. IEEE Transactions on Fuzzy Systems. Volume 15, Number 4, August 2007.
- B. Carse, A.G. Pipe. Special Issue on Genetic Fuzzy Systems. International Journal of Intelligent Systems. Volume 22, Number 9, September 2007.
- J. Casillas, B. Carse. Special Issue on Genetic Fuzzy Systems: Recent Developments and Future Directions. Soft-Computing Volume 13, Number 5, March 2009.

The collection of papers that we could find on these special issues give us a historical tour on the different stages we can find in the evolution of GFSs research:

- The two first special issues (1997, 1998) contain contributions devoted to learning KB components using the different learning approaches (Michigan, IRL, Pittsburgh) together with some applications. We can find representative approaches of different areas of the taxonomy.
- In the next two special issues (2001, 2004) we can find contributions that exploit the mentioned genetic learning approaches together with contributions that stress new branches such as genetic rule selection, multiobjective genetic algorithms for rule selection, the use of genetic programming for learning fuzzy systems, hierarchical genetic fuzzy systems, coevolutionary genetic fuzzy systems, the combination of boosting and evolutionary fuzzy systems learning, embedded genetic DB learning, and first studies for dealing with high dimensional problems, among others. We would like to point out the review paper that was published in the last issue [36] that was the first review in the topic, briefly introducing GFS models and applications, trends and open questions. Another short review was presented in [72]. The present chapter can be considered as a continuation of those, with the novelty of the taxonomy, the GFSs outlook based on the pioneer papers, the ISI Web of Science based visibility and the milestones along the GFSs history and new trends and prospects.
- The next three special issues, published in 2007, emphasize three different directions. Carse and Pipe's special issue collect papers focused in the mentioned areas (multiobjective evolutionary learning, boosting and evolutionary learning, etc) and stress some new ones such as evolutionary adaptive inference systems. Casillas et al.'s special issue is focused on the trade-off between interpretability and accuracy, collecting four papers that proposed different GFSs for tackling this problem. Cordón et al.'s special issue focuses its attention on novel GFS proposals under the title "What's Next?", collecting highly innovative GFS proposals that can mark new research trends. The four collected papers are focused on: a new Michigan approach for learning RBs based on XCS [22], GFSs for imprecisely observed data (low quality data) [162], incremental evolutionary learning of TS-fuzzy systems [86], and evolutionary fuzzy rule induction for subgroup discovery [48].
- The last special issue, co-edited by J. Casillas and B. Carse, is devoted to new developments, paying attention to multiobjective genetic extraction of linguistic fuzzy rule based systems from imprecise data [163], multiobjetive genetic rule selection and tuning [60], parallel distributed genetic fuzzy rule selection [144], context adaptation of fuzzy systems [17], compact fuzzy systems [28], neuro-coevolutionary GFSs [153], evolutionary learning of TSK rules with variable structure [140] and genetic fuzzy association rules extraction [29].

3.4 Current Research Trends in GFSs

In this subsection, from the abundant GFSs literature published, we focus our attention into six current trends that are of high interest at the present and show considerable potential in the near future.

Evolutionary Multiobjective learning of FRBSs: interpretability-precision trade-off

Multiobjective evolutionary algorithms (MOEAs) are one of the most active research areas in the field of evolutionary computation, due to population-based algorithms being capable of capturing a set of non-dominated solutions in a single run of the algorithm. A large number of algorithms have been proposed in the literature [45, 34]. Among them, NSGA-II [46] and SPEA2 [209] are well known and frequently-used MOEAs.

Obtaining high degrees of interpretability and accuracy is a contradictory aim, and, in practice, one of the two properties prevails over the other. Nevertheless, a new tendency in the fuzzy modelling scientific community that looks for a good balance between interpretability and accuracy is increasing in importance. The improvement of the interpretability of rule based systems is a central issue in recent research, where not only the accuracy is receiving attention but also the compacting and the interpretability of the obtained rules [114, 138].

In multiobjective GFSs it is desirable to design genetic learning algorithms in which the learning mechanism itself finds an appropriate balance between interpretability and accuracy. We consider objectives based on accuracy and objectives that include different complexity/interpretability measures. Figure 14 from [91] illustrates this idea where each ellipsoid denotes a fuzzy system. There exists a large number of non-dominated fuzzy systems along the accuracy-complexity trade-off curve.

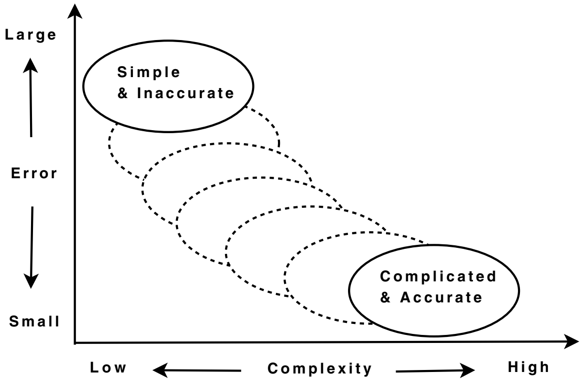


Fig. 14 Non-dominated fuzzy systems

There exists an important number of contributions focused on this topic, in fact, Chapter 5 of this book is devoted to this topic. Therefore, we will not extend our description on the topic.

GA-based techniques for mining fuzzy association rules and novel data mining approaches

Fayyad et al. defined knowledge discovery (KD) as the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [57]. KD may not be viewed as synonymous with DM, but they are intimately related. KD is a wide ranging process which covers distinct stages: the comprehension of the problem, the comprehension of the data, pre-processing (or preparation) of the data, DM and post-processing (assessment and interpretation of the models). The DM stage is responsible for automatic KD at a high level, and from information obtained from real data. Some of the important problems that DM and KD deal with are: rule extraction, identification of associations, feature analysis, linguistic summarization, clustering, classifier design and novelty/anomaly detection.

The interpretability of knowledge is crucial in the field of DM/KD where knowledge should be extracted from data bases and represented in a comprehensible form, or for decision support systems where the reasoning process should be transparent to the user. In fact, the use of linguistic variables and linguistic terms in a discovered process has been explored by different authors.

Frequent pattern mining has been a focused theme in DM research for over a decade. Association analysis is a methodology that is useful for the discovery of interesting relationships hidden in large data sets. The uncovered relationships can be represented in the form of association rules or sets of frequent items. Abundant literature can be found presenting tremendous progress in the topic [179, 71].

As claimed in [54], the use of fuzzy sets to describe associations between data extends the types of relationships that may be represented, facilitates the interpretation of rules in linguistic terms, and avoids unnatural boundaries in the partitioning of the attribute domains.

Linguistic variables with linguistic terms can contribute in a substantial way to the advance in the design of association rules and the analysis of data to establish relationships and identify patterns, in general [90]. On the other hand, GAs in particular, and EAs in general, are widely used for evolving rule extraction and patterns association in DM/KD [59]. The conjunction in the GFS field provides novel and useful tools for pattern analysis and for extracting new kinds of useful information with a distinct advantage over other techniques: its interpretability in terms of fuzzy IF-THEN rules. We find interesting recent contributions focused on the genetic extraction of fuzzy association rules in [102, 89, 101, 184].

We would like to pay attention to a subdivision of descriptive induction algorithms which has recently received attention from researchers, called subgroup discovery. It is a form of supervised inductive learning of subgroup descriptions in which, given a set of data and having a property of interest to the user, attempts to locate subgroups which are statistically “most interesting” for the user. Subgroup discovery has the objective of discovering interesting properties of subgroups obtaining simple rules (i.e. with an understandable structure and with few variables), highly significant and with high support (i.e. covering many of the instances of the target class). The concept was initially formulated by Klösgen in his rule learning algorithm EXPLORA [108] and by Wrobel in the algorithm MIDOS [201]. Both use a rule-extraction model based on decision trees, in order to obtain the best subgroups among the population. In order to evaluate the subgroups, evaluation measurements are defined which determine the interest of an expression through a combination of unusualness and size. MIDOS tackles, within this same approach, the problem of discovery in multi-relational databases. A recent study on the topic can be found in [118]. In [48] we find a first approach to the use of GFSs for subgroup discovery.

The use of GFSs for association analysis is a topic that would provide interesting future contributions focusing attention on the different research problems that we can find in the frequent pattern mining area [71].

Learning genetic models based on low quality data (noise data and vague data)

There are many practical problems requiring learning models from uncertain data. The experimental designs of GFSs learning from data observed in an imprecise way are not being actively studied by researchers. However, according to the point of view of fuzzy statistics, the primary use of fuzzy sets in classification and modelling problems is for the treatment of vague data. Using vague data to train and test GFSs we could analyze the performance of these classifiers on the type of problems for which fuzzy systems are expected to be superior. Preliminary results in this area involve the proposals of different formalizations for the definition of fuzzy classifiers, based on the relationships between random sets and fuzzy sets [161] and the study of fitness functions (with fuzzy values) defined in the context of GFSs [162].

This is a novel area that is worth being explored in the near future, which may provide interesting results.

Genetic learning of fuzzy partitions and context adaptation

The DB learning comprises the specification of the universes of discourse, the number of labels for each linguistic variable, as well as the definition of the fuzzy membership functions associated with each label. In [39] the influence of fuzzy partition granularity in the FRBS performance was studied. Showing that using an appropriate number of terms for each linguistic variable, the FRBS accuracy can be significantly improved without the need of a complex RB learning method.

On the other hand, the idea of introducing the notion of context into fuzzy systems comes from the observation that, in real life, the same basic concept can be perceived differently in different situations. In some cases, this information is related to the physical properties or dimensions of the system or process, including restrictions imposed due to the measurement acquisition or actuators. In the literature, context adaptation in fuzzy systems has been mainly approached as the scaling of fuzzy sets from one universe of discourse to another by means of non-linear scaling functions whose parameters are identified from data.

Different approaches have been proposed to deal with the learning of membership functions, granularity, non-linear contexts using GAs, etc. [133, 69, 40, 41, 15, 16, 6].

Although there is a large number of contributions in the area of DB Learning, we think that this remains a promising research area, due to the importance of using adequate membership functions and an appropriate context. The use of GFSs has much potential due to its flexibility for encoding DB components together with other fuzzy system components.

Genetic adaptation of inference engine components

We know that it is possible to use parametric aggregation operators in the design of the inference system and the defuzzification method, in an attempt to get the most appropriate parameter configuration in any application. The tuning of these components can be considered to get more accurate fuzzy models. We have come across different GFS approaches for finding the most appropriate parameters [42, 8].

This is an interesting research area that can provide us with the opportunity to adapt the inference parameters to an FRBS and to design learning models that can coevolve the inference engine parameters together with the KB components.

Revisiting the Michigan-style GFSs

The first description of a Michigan-style GFS was given in [186]. All the initial approaches in this area were based on the concept of“rule strength” in the sense that a rule (classifier) gains “strength” during interactions with the environment (through rewards and /or penalties). This strength can then be used for two purposes: resolving conflicts between simultaneously matched rules during learning episodes; and as the basis of fitness for the GAs.

A completely different approach can be considered in which a rule’s fitness, from the point of view of the GA, is based on its “accuracy”, i.e., how well a rule predicts payoff whenever it fires. Notice that the concept of accuracy used here is different from that traditionally used in fuzzy modelling (i.e., the capability of the fuzzy model to faithfully represent the modelled system). This accuracy-based approach offers a number of advantages, such as avoiding overgeneral rules, obtaining optimally general rules, and learning a complete covering map. The first accuracy-based evolutionary algorithm, called XCS, was proposed in [199] and it is currently of major interest to the research community in this field.

Casillas et al. proposed in [22] a new approach to achieve accuracy-based Michigan-style GFSs. The proposal, Fuzzy-XCS, is based on XCS but properly adapted to fuzzy systems, with promising results for function approximation problems and for robot simulation online learning. In [145], an extension of the UCS algorithm is proposed, a recent Michigan-style genetic learning algorithm for classification [14].

These approaches build a bridge between the Michigan-style genetic learning studies and the fuzzy systems models. This is a promising research line that can provide interesting results in the near future.

4 Fuzzy Evolutionary Algorithms

Nowadays, there exists an increasing interest in the use of fuzzy tools and fuzzy logic-based techniques for modelling different EA components or adapting EA control parameters, with the aim of enhancing the performance of these search algorithms [79, 149, 183]. Generally, EAs resulting from this integration are called *fuzzy* EAs.

This section focuses on fuzzy EAs. We give an overview of the existing research on this topic, describing several instances grouped into three categories that were identified after revising specialized literature. The first one involves the adaptation of GA control parameters by means of FRBSs (in particular, fuzzy logic controllers) and, at present, it has a consolidated background of knowledge (Section 4.1). The second one includes those EA models whose components (genetic operators, representation, stop criterion, etc.) are designed using fuzzy tools (Section 4.2). The third one consists of different innovative EA models (particle swarm optimization algorithms, ant colony optimization algorithms, differential evolution, etc.) that make use of fuzzy logic as way to improve their performance (Section 4.3). In addition, we attempt to identify some open issues and summarize a few new promising research directions for fuzzy EAs (Section 4.4).

4.1 Fuzzy Adaptive GAs

Adaptive GAs dynamically adjust their parameters during the course of evolving a solution with the aim of inducing exploitation/exploration relationships that avoid the premature convergence problem and improve the final results [185, 55]. However, the design of this type of GA is very difficult, because the interaction of GA control parameter settings and GA performance is generally acknowledged as a complex relationship which is not completely understood. Although there are ways to understand this relationship (for instance, in terms of stochastic behavior), this kind of understanding does not necessarily result in a normative theory.

Fuzzy logic controllers (FLCs) [53] are a particular model of FRBS (Section 2) that provide a tool which can convert the linguistic control strategy based on expert knowledge into an automatic control strategy. They are particularly suited to model the relationship between variables in environments that are either ill-defined or very complex.

The adaptation of GA parameters is one such complex problem that may benefit from the use of FLCs, producing the so-called fuzzy adaptive GAs (FAGAs) [78, 123]. The rule-bases of FLCs facilitate the capture and representation of a broad range of adaptive strategies for GAs (for example, they may provide the support for the automatic learning of such strategies). The main idea of FAGAs is to use an FLC whose inputs are any combination of GA performance measures or current control parameters and whose outputs are GA control parameters. Current performance measures of the GA are sent to the FLC, which computes new control parameter values that shall be used by the GA. Figure 15 shows this process.

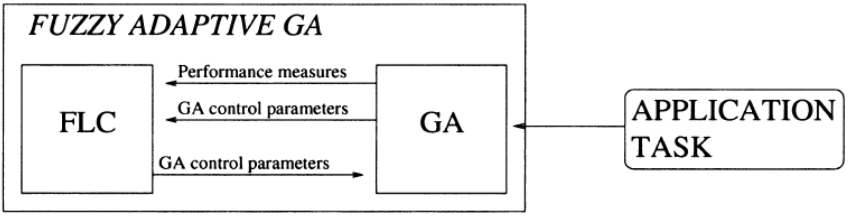


Fig. 15 FAGA model

4.1.1 Designing FAGAs

In this section, we briefly describe the issues that should be tackled in order to build the FLC used by an FAGA. They include the choice of inputs and outputs, the definition of the data base associated with them, and the specification of the rule-base:

Inputs, Outputs, and Data Base

- *Inputs.* They should be robust measures that describe GA behaviour and the effects of the genetic setting parameters and genetic operators. Some possible inputs may be: diversity measures, maximum, average, and minimum fitness, etc. The current control parameters may also be considered as inputs.
- *Outputs.* They indicate values of control parameters or changes in these parameters. In [182], the following outputs were reported: mutation probability, crossover probability, population size, selective pressure, the time the controller must spend in a target state in order to be considered successful, the degree to which a satisfactory solution has been obtained, etc.
- *Data Base.* Each input and output should have an associated set of linguistic labels. The meaning of these labels is specified through membership functions of fuzzy sets, the fuzzy partition, contained in the Data Base. Thus, it is necessary that every input and output have a bounded range of values in order to define these membership functions over it.

Rule-Base

After selecting the inputs and outputs and defining the Data Base, the fuzzy rules describing the relations between them should be defined. They facilitate the capture and representation of a broad range of adaptive strategies for GAs.

Although, the experience and the knowledge of GA experts may be used to derive rule-bases, many authors have found difficulties in doing this. In this sense, the following three reflections were quoted by different authors:

> *"Although much literature on the subject of GA control has appeared, our initial attempts at using this information to manually construct a fuzzy system for genetic control were unfruitful."* [120].

> *"Statistics and parameters are in part universal to any evolutionary algorithm and in part specific to a particular application. Therefore it is hard to state general fuzzy rules to control the evolutionary process."* [182].

> *"The behaviour of GAs and the interrelations between the genetic operators are very complex. Although there are many possible inputs and outputs for the FLCs, frequently fuzzy rule-bases are not easily available: finding good fuzzy rule bases is not an easy task."* [74].

Automatic learning mechanisms to obtain rule-bases have been introduced to avoid this problem. By using these mechanisms, relevant relations and membership functions may be automatically determined and may offer insight to understand the complex interaction between GA control parameters and GA performance [120]. Two types of rule-base learning techniques have been presented: the *offline* learning technique [120, 121] and the *online* learning technique [77]:

- The *offline* learning mechanism is an evolutionary algorithm that is executed once, before the operation of the FAGA, however it has associated with it a high computational cost. It works by considering a fixed set of test functions, following the same idea as the meta-GA of Grefenstette [68]. Unfortunately, the test functions may have nothing to do with the particular problem to be solved, which may limit the robustness of the rule-bases returned.
- In the *online learning* technique, the rule-bases used by the FLCs come from an evolutionary process that interacts concurrently with the GA to be adapted. The learning technique underlying this approach only takes into account the problem to be solved (in contrast to the previous one, which never considers it). In this way, the rule-bases obtained will specify adaptation strategies particularly appropriate for this problem.

4.1.2 A Taxonomy for FAGAs

In this section, we present a taxonomy for FAGAs, focussing on the combination of two aspects:

- The way in which the rule-bases are derived:
  - Through the *expertise, experience*, and *knowledge* of GAs, which have become available as a result of empirical studies conducted over a number of years.
  - Using an *offline learning mechanism*, which finds rule-bases that induce a suitable FAGA behaviour on a fixed set of test functions. It is executed before the application of the FAGA on any real problem.
  - By means of an *online learning mechanism*, which learns rule-bases during the run of the FAGA on a real problem.
- The level where the adaptation takes place in FAGAs:
  - *Population-level* adaptations adjust control parameters that apply for the entire population.

- *Individual-level* adaptations tune control parameters that have an effect on the individual members of the population.

Table 1 outlines the main features of several FAGA instances presented in the literature. It includes the inputs and outputs of the FLCs, the adaptation level, and the method considered to derive the rule-base. A visual inspection of Table 1 allows one to conclude that:

1. The study of FAGAs has been an active line of research in the evolutionary computation community that has produced a significant amount of work during the last fifteen years.
2. Most FAGAs presented in the literature involve population-level adaptation. However, adaptive mechanisms at the individual level based on FLCs may be interesting to adjust control parameters associated with genetic operators [210, 77]. In this case, the control parameters will be defined on individuals instead of on the whole population. Inputs to the FLCs may be central measures and/or measures associated with particular chromosomes or sets of them, and outputs may be control parameters associated with genetic operators that are applied to those chromosomes. A justification for this approach is that it allows for the application of different search strategies in different parts of the search space. This is based on the reasonable assumption that, in general, the search space will not be homogeneous, and that different strategies will be better suited to different kinds of sublandscapes.
3. Most instances use rule-bases derived from GA experts. The use of an online learning mechanism has been less explored, though nowadays it is becoming one of the most prospective alternatives (see Section 4.4.1). An example of is approach was proposed in [77], which was called *coevolution with fuzzy behaviours*. Its main ideas are:

   - It incorporates genetic operator adaptation at an individual-level based on FLCs. Control parameter values for a genetic operator are computed for each set of parents that undergo it, using an FLC that considers particular features associated with the parents as inputs.
   - The rule-bases of the FLCs applied are learnt implicitly throughout the run by means of a separate GA that *coevolves* with the one that applies the genetic operator to be controlled. The goal of this GA is to obtain the rule-bases that produce suitable control parameter values to allow the genetic operator to show an adequate performance on the particular problem to be solved.

   Since the learning technique underlying this approach only takes into account the problem to be solved (in contrast to the approaches based on offline learning mechanisms), the rule-bases obtained shall specify adaptation strategies particularly appropriate for this problem.

Table 1 Instances of FAGAs in the literature

FAGA Instances	Inputs	Outputs	Adaptation Level	Method to Derive Rule-Base
Xu and Vukovich (1993, 1994) [202, 203]	Generation and population size	p_c and p_m	Population-level	GA expert knowledge
Lee and Takagi (1993, 1994) [120, 121]	Two phenotypical diversity measures and change in the best fitness since the last control action	Changes to p_c and p_m, and population size	Population-level	Offline learning
Bergmann, Burgard, and Hemker (1994) [13]	Entropy evolution	Inversion rate, p_c, and p_m	Population-level	GA expert knowledge
Herrera and Lozano (1996) [74]	Genotypical diversity measure and phenotypical diversity measure	Frequency of application of two crossover operators and selection pressure	Population-level	GA expert knowledge
Wang et al (1996) [198]	Change in average fitness of the population at two consecutive generations	Changes to p_c and p_m	Population-level	GA expert knowledge
Zeng and Rabenasolo (1997) [210]	Variance of fitness values, distance between the fitness of the best parent and the best fitness, distance between parents, and normalized fitness values of the parents	p_c for every pair of parents	Individual-level	GA expert knowledge
Song et al. (1996, 1997) [172, 173]	Change in average fitness of the population at two consecutive generations	Changes to p_c and p_m	Population-level	GA expert knowledge
Clintock, Lunney, and Hashim (1997) [31, 32]	Population statistics and diversity statistics	p_c, p_m, and parameter that determines the application of different crossover operators	Population-level	GA expert knowledge
Subbu, Sanderson, and Bonissone (1998) [175]	Genotypic and phenotypic diversity measures of the population	Population size, p_c, and p_m	Population-level	Offline learning
Shi, Eberhart, and Chen (1999) [168]	Best fitness, number of generations for unchanged best fitness, and variance of fitness	p_c and p_m	Population-level	GA expert knowledge
Herrera and Lozano (2000) [76]	Current pm and convergence measure	p_m	Population-level	GA expert knowledge
Matousek, Osmera, and Roupec (2000) [136]	Variability of population, coefficient of partial convergente, and H-characteristics	p_m and selection pressure	Population-level	GA expert knowledge
Wang (2001) [196]	Genetic drift degree, phenotypical diversity measure and number of generations without improving the best individual	p_c and p_m	Population-level	GA expert knowledge
Herrera and Lozano (2001) [77]	Ranks associated with the parents with regards to their fitness values in the population	Control parameter associated with fuzzy recombination	Individual-level	Online learning
Zhu, Zhang, and Jing (2003) [211]	Population size, generation number, and two phenotypic measure for both diversity and convergence	p_c, p_m, and selection pressure	Population-level	Online learning
Yun and Gen (2003) [207]	Changes of average fitness in population of two continuous generations	Changes to p_c and p_m	Population-level	GA expert knowledge
Subbu and Bonissone (2003) [176]	Genotypic diversity and percentage completed trials	Changes to the population size and p_m	Population-level	GA expert knowledge
Ah King, Radha, and Rughooputh [1]	Change in average fitness of the population at two consecutive generations	Changes to p_c and p_m	Population-level	GA expert knowledge
King, Radha, and Rughooputh (2004) [106]	Changes in average fitness at two consecutive steps	Changes to p_c and p_m	Population-level	GA expert knowledge
Last and Eyal (2005, 2006). [115, 116]	Age and lifetime of the chromosomes to be crossed over (parents) and the population average lifetime	p_c	Individual-level	GA expert knowledge
Liu, Xu, and Abraham (2005) [126]	Changes of the best fitness and average fitness in the GA population of two continuous generations	Changes to p_c and p_m	Population-level	GA expert knowledge
Li et al (2006) [122]	Average fitness value of the individuals and standard deviation between two consecutive generations	p_c and p_m	Population-level	GA expert knowledge
Hamzeh, Rahmani, and Parsa (2006) [70]	Measures associated with an XCS learning classifier system	Exploration probability rate	Population-level	GA expert knowledge
Lau, Chan, and Tsui (2007) [117]	Average fitness values in the population and measure of population diversity	Changes to p_c and p_m	Population-level	GA expert knowledge
Sahoo et al (2006, 2007) [159, 160]	Standard deviation of fitness distribution of population and incremental change in average fitness of the population from generation to generation	p_m	Population-level	GA expert knowledge

4.2 EA Components Based on Fuzzy Tools

In this section, we review different EA components built using fuzzy tools that have appeared in the literature.

Fuzzy Genetic Operators

Fuzzy connectives and triangular probability distributions have been considered for designing powerful real-parameter crossover operators that establish adequate population diversity levels and thus help to avoid premature convergence:

- *FCB-crossovers* [82]. These are crossover operators for real-coded GAs based on the use of fuzzy connectives: t-norms, t-conorms and average functions. They were designed to offer different exploration and exploitation degrees.
- *Heuristic FCB-crossovers* [75]. These produce a child each whose components are closer to the corresponding component of its fitter parent.
- *Dynamic FCB-crossovers* [81]. These are crossover operators based on the use of parameterized fuzzy connectives. These operators keep a suitable sequence between the exploration and the exploitation along the GA run: *"to protect the exploration in the initial stages and the exploitation later"*.
- *Dynamic Heuristic FCB-crossovers* [81]. These operators put together the heuristic properties and the features of the Dynamic FCB-crossover operators. They showed very good results as compared with other crossover operators proposed for RCGAs, even better than the FCB-crossover operators and the dynamic ones.
- *Soft Genetic Operators.* In [192, 193, 195], crossover and mutation operators were presented, which are based on the use of triangular probability distributions. These operators, called *soft modal* crossover and mutation, are a generalization of the discrete crossover operator and the BGA mutation, respectively, proposed for the *Breeder* GA [141]. The term *soft* is gleaned from fuzzy set theory only to help grasp the main idea, since probability distributions are considered instead of membership functions.

Fuzzy Representations

Classical EAs, such as GAs and evolution strategies, do not take into account the *development* of an individual or organism from the gene level to the mature phenotype. There are no one-gene, one-trait relationships in natural evolved systems. The phenotype varies as a complex, non-linear function of the interaction between underlying genetic structures and current environmental conditions. Nature follows the universal effects of *pleiotropy* and *polygeny*. Pleiotropy is the fact that a single gene may simultaneously affect several phenotype traits. Polygeny is the effect when a single phenotypic characteristic may be determined by the simultaneous interaction of many genes [58]. An attempt to deal with more complex genotype/phenotype relations in EAs was presented in [191, 194]. A *fuzzy representation* is proposed for the case of tackling optimization problems of parameters with variables on continuous domains. Each problem parameter has associated a number (m) *fuzzy decision* *variables* belonging to the interval $[0,1]$. The chromosomes are formed by linking together the values of the decision variables for each parameter. For each parameter, the decoding process is carried out using a function $g : [0,1]^m \rightarrow [0,1]$, and a linear transformation from the interval [0,1] to the corresponding parameter domain. As an example of such a function the authors presented the following:

$$\forall d = (d_1, ..., d_m) \in [0,1]^m, \quad g(d) = \frac{1}{2^{m-1}-1} \sum_{j=1}^{m} d_j 2^{j-1}.$$

When $m > 1$, this coding type breaks the one-to-one correspondence between genotype and phenotype (assumed by classical EAs), since two different genotypes may induce the same phenotype. So, it is impossible to find inferences from phenotype to genotype, i.e., the mapping from genotype to phenotype is not *isomorphic*. Different experiments carried out in [194] with $m = 1$ and $m = 2$ showed that the use of a fuzzy representation allows robust behavior to be obtained. In some cases, a better performance than the Breeder GA was achieved. Furthermore, another important conclusion was stated: for a small population size the performance for $m = 2$ is slightly better than for $m = 1$, whereas the opposite is true for large population sizes.

Sharma and Irwin [167], addressed the use of appropriate fuzzy sets to represent a parameter depending upon its contribution within a problem domain. They proposed a chromosome encoding method, named *fuzzy coding*, for representing real number parameters in a GA. Fuzzy coding is an indirect method for representing a chromosome, where each parameter is represented by two sections. In the first section, the fuzzy sets associated with each parameter are encoded in binary bits with a "1" representing the corresponding set selected. In the second section, each parameter contains degrees of membership corresponding to each fuzzy set. These are encoded as real numbers and represent the degrees of firing. The actual parameter value of interest is obtained through the information contained in the chromosome by means of a defuzzification method. This coding method represents the knowledge associated with each parameter and is an indirect method of encoding compared with the alternatives in which the parameters are directly represented in the encoding. Two test examples, along with neural identification of a nonlinear *pH* (measure of acidity or alkalinity of water) process from experimental data, were studied. It was shown that fuzzy coding is better than the conventional methods (binary, gray, and floating-point coding) and is effective for parameter optimization in problems where the search space is complicated. In addition, the authors claim that this new technique also has the flexibility to embed prior knowledge from the problem domain which is not possible in the regular coding methods. We should point out that an additional investigation was carried out by Pedycz [149] into the exploitation of fuzzy sets as a basis for encoding an original search space.

Finally, in [174], an algorithm for adaptively controlling GA parameter coding using fuzzy rules is presented, which was called fuzzy GAP. This uses an intermediate mapping between the genetic strings and the search space parameters. In particular, each search parameter is specified by the following equation:

$$p_s = (\frac{p_g}{2^l - 1}) \cdot R + O,$$

where p_s is the search parameter, p_g is the genetic parameter, l is the number of bits in the genetic parameter, R is a specified parameter range, and O is a specified offset. By controlling the offset and range, more accurate solutions are obtained using the same number of binary bits.

Fuzzy GAP performs a standard genetic search until the population of strings has converged. Convergence was measured by evaluating the average number of bits which differ between all the genetic strings. Each string is compared to every other string and the number of different bits is counted. If the average number of differing bits per string pair is less than a threshold, the GA has converged. After the genetic strings have converged, a new range and offset for the search parameters are determined by means of an FLC with an input that measures the distance between the centre of the current range and the best solution found in the search. After applying the FLC, the GA is executed again with the new values for the range and offset. The performance of fuzzy GAP on a hydraulic brake emulator parameter identification problem was investigated. It was shown to be more reliable than other dynamic coding algorithms (such as the dynamic parameter encoding algorithm), providing more accurate solutions in fewer generations.

Fuzzy Stopping Criteria

Due to the possibility of premature convergence, GAs do not guarantee that the optimal solution shall be found. Therefore, if the optimal solution is not known, GA performance is difficult to measure accurately. In [137], a fuzzy stopping criterion mechanism (FSCM) is developed to provide a useful evaluation of the GA's real time performance. FSCM is based on achieving a user-defined level of performance for the given problem. In order to do so, it includes a predicting process based on statistics for estimating the value of the GA optimal solution, then it compares the current solution to this optimal one by checking if an acceptable percentage (specified by the user) of the latter is reached. If so, the GA stops and returns belief and uncertainty measures that provide reliability measure for the GA chosen solution. The acceptable percentage optimal solution defined by the user represents a fuzzy stopping criterion for halting GA if an appropriate solution is reached. The predicting process is invoked every 40 iterations and uses performance values such as the minimum solution value, average solution value and belief and plausibility values, all obtained during these iterations. The underlying idea for the FSCM is that the user does not need to find the global solution, but rather an approximate solution that is close to the optimal one, i.e., the GA is used for solving a *fuzzy goal* instead of a crisp one because of the vagueness of the term approximate. This term is quantitatively measured by the user through the acceptable percentage of the optimal solution that he requires in the final solution. Results obtained on a 25-city TSP problem indicate this approach is preferable to a simple GA, in term of cost/performance and in decreasing the amount of time the GA searches for acceptable solutions.

4.3 Other Fuzzy EA Models

Different fuzzy logic tools have been employed to improve the behavior of other EA models, such as EAs for multiobjective problems, parallel EAs, genetic programming, differential evolution, particle swarm optimization algorithms, ant colony optimization algorithms, and cultural algorithms. Next, we briefly explain the way these EA approaches benefited from fuzzy logic.

Fuzzy EAs for Multiobjective Optimization Problems

In [189], a FAGA is presented for multiobjective optimization problems. In each generation, an FLC decides what transformation of the cost components into a one-dimensional fitness function is taken. In this vein, [152], Rachmawati and Srinivasan present an algorithm that employs a fuzzy inference system to model and aggregate different objectives. They are represented as fuzzy variables, which act as inputs to a fuzzy inference system evaluating the fitness of the associated candidate solution. The fuzzy system captures preferences of the decision maker in the compromise between various objectives, thereby guiding the search to interesting regions in the objective space. In [190], a more complex method, called a *fuzzy reduction* GA, is proposed. It attempts to enable a uniform approximation of the Pareto optimal solutions (those that cannot be improved with respect to any cost function without making the value of some other worse). The authors started by explicitly formulating desirable goals for the evolution of the population towards the target Pareto optimal solutions (which could be expressed in vague terms only). Then, they defined deviation measures for a population from these goals, which were the inputs to an FLC. Later, they fixed a set of possible actions that could serve as countermeasures to decrease the deviations. These actions are different selection mechanisms based on classical ones proposed to tackle multiobjective optimisation problems. The FLC determines activation rates for the actions. The action that should actually be taken is decided according to the activation rates found. As an application, a timetable optimisation problem is presented where the method was used to derive cost-benefit curves for the investment into railway nets. The results showed that the fuzzy adaptive approach avoids most of the empirical shortcomings of other multiobjective GAs by the adaptive nature of the procedure. Other models of multiobjective GA based on the fuzzy logic tools are found in [44, 52, 98, 119].

Fuzzy Parallel EAs

The availability, over the last few years, of fast and cheap parallel hardware has favoured research into possible ways for implementing parallel versions of EAs [20]. EAs are good candidates for effective parallelization, since they are inspired on the principles of parallel evolution, for a population of individuals. Among the many types of parallel EAs, *distributed* and *cellular* algorithms are two popular optimization tools. The basic idea of the distributed EAs lies in the partition of the population into several subpopulations, each one of them being processed by an EA, independently from the others. Furthermore, a migration process produces a chromosome exchange between the subpopulations. An important control parameter that determines the operation of this process is the migration rate, which controls how many chromosomes migrate. Maeda et al. [131, 132] propose an adaptive search method for distributed EAs. Its main characteristic feature is the fuzzy adaptive control of the migration rate by evaluating the evolutionary degree for each subpopulation. Simulations were performed to confirm the efficiency of this method, which was shown to be superior to both ordinary and parallel EAs. In a cellular EA, the concept of (small) neighbourhood is intensively used; this means that an individual may only interact with its nearby neighbours in the breeding loop [3]. In [156], the fuzzy adaptive mechanism proposed by Shi et al. [168] was considered to adapt parameters associated with cellular EAs, obtaining a fuzzy cellular EA model.

Fuzzy Genetic Programming

Genetic Programming's [111] basic distinction from GAs is the evolution of dynamic tree structures, often interpreted as programs, rather than fixed-length vectors. In [10], it is claimed that genetic programming requires human supervision during their routine use as practical tools for the following reasons: 1) to detect the emergence of a solution, 2) to tune algorithm parameters and 3) to monitor the evolution process in order to avoid undesirable behaviour such as premature convergence. It is also advised that any attempt to develop artificial intelligence tools based on genetic programming should take these issues into account. The authors proposed FLCs for this task. They called the collection of fuzzy rules and routines in charge of controlling the evolution of the GA population "fuzzy government". Fuzzy government was applied to the symbolic inference of the formulae problem. Genetic programming was used to solve the problem along with different FLCs, which dynamically adjusted the maximum length for genotypes, acted on the mutation probability, detected the emergence of a solution, and stopped the process. The results showed that the performance of the fuzzy governed GA was almost impossible to distinguish from the performance of the same algorithm operated directly with human supervision. Other work on fuzzy adaptive search methods for genetic programming is [130].

Fuzzy Cultural Algorithms

Cultural algorithms (CAs) [154] are dual inheritance systems that consist of a social population and a belief space. The problem solving experience of individuals selected from the population space by an acceptance function is used to generate problem solving knowledge that resides in the belief space. This knowledge can be viewed as a set of beacons that can control the evolution of the population component by means of an influence function. The influence function can use the knowledge in the belief space to modify any aspect of the population component. Various evolutionary models have been used for the population component of CAs, including GAs, genetic programming, evolution strategies, and evolutionary programming. In [155], a fuzzy approach to CAs is presented in which an FLC regulates the amount of information to be transferred to the belief space used by the CA over time. In particular, the FLC determines the number of individuals which shall impact the current beliefs. Its inputs are the individual success ratio (ratio of the number of successes to the total number of mutations) and the current generation. A comparison was made between the fuzzy version of a CA (that used evolutionary programming as the population component) and its non fuzzy version on 34 optimization functions. The conclusions were: 1) the fuzzy interface between the population and belief space outperformed the non fuzzy version in general, and 2) the use of a fuzzy acceptance function significantly improved the success ratio and reduced CPU time.

Fuzzy Ant System

Ant Colony Optimization (ACO) [51] is a population-based metaheuristic approach for solving hard combinatorial optimization problems. The inspiring source of ACO is the foraging behavior of real ants which enables them to find shortest paths between a food source and their nest. They are based on a colony of artificial ants, that is, simple computational agents that work cooperatively and communicate through artificial pheromone trails. ACO algorithms are essentially construction algorithms: every ant constructs a solution to the problem by travelling on a construction graph. The edges of the graph, representing the possible steps the ant can make, have two kinds of associated information (heuristic information and artificial pheromone trail information), which are used to define transition probabilities of moving from one node to other, guiding ant movement. This information is modified during the algorithm run, depending on the solutions found by the ants. In [181], a fuzzy ACO approach is presented, which uses fuzzy logic to calculate an ant's utility to visit the next node. In particular, transition probabilities (usually given in a classical ACO in closed form) are computed by a fuzzy rule-based system. Their authors claim that when using fuzzy logic as a separate module within the ACO, it is possible to handle the uncertainty that sometimes exists in some complex combinatorial optimization problems. The control strategies of an ant can also be formulated in terms of descriptive fuzzy rules. Other ACO models based on fuzzy logic are presented in [104, 142].

Fuzzy Particle Swarm Optimization

Particle Swarm Optimization (PSO) algorithm [103] is inspired by social behaviour patterns of organisms that live and interact within large groups. In particular, PSO incorporates swarming behaviours observed in flocks of birds, schools of fish, or swarms of bees, and even human social behaviour. The standard PSO model consists of a swarm of particles, which are initialized with a population of random candidate solutions. Each particle has a position represented by a position-vector, and a velocity represented by a velocity vector. The particles move iteratively through the d-dimension problem space to search new solutions, where the fitness can be calculated as a quality measure. A particle decides where to move next, considering its own experience, which is the memory of its best past position, and the experience of the most successful particle in the swarm. It has been shown that the trajectories of the particles oscillate in different sinusoidal waves and converge quickly, sometimes prematurely. Liu and Abraham [124] proposed an adaptive mechanism based on FLCs to control the velocity of particles in order to avoid premature convergence in PSO. Empirical results demonstrated that the performance of standard PSO degrades remarkably with the increase in the dimension of the problem, while the influence is very little in the fuzzy PSO approach. Another instance of a PSO model tuned by FLCs may be found in [99]. Finally, we should point out that a fuzzy version of PSO specifically designed to tackle the quadratic assignment problem was presented in [125].

Fuzzy Differential Evolution

The differential evolution algorithm (DE) is one of the most recent EAs for solving real-parameter optimization problems [151]. Like other EAs, DE is a population-based, stochastic global optimizer capable of working reliably in nonlinear and multimodal environments. DE has few control parameters. However, choosing the best parameter setting for a particular problem is not easy [129]. Liu and Lampinen [127, 128, 129] present the fuzzy adaptive differential evolution algorithm, which uses FLCs controllers to adapt the search parameters for the DE mutation operation and crossover operation. These two parameters were adapted individually for each generation. Parameter vector change and function value change over the whole population members between the last two generations were nonlinearly depressed and then used as the inputs for both FLCs. Experimental results, provided by the proposed algorithm for a set of standard test functions, outperformed those of the standard differential evolution algorithm for optimization problems with higher dimensionality.

4.4 Future Work on Fuzzy EAs

Despite the recent activity and the associated progress in fuzzy EAs research, there remain many directions in which the work may be improved or extended. Next, we report on some of these.

4.4.1 Improvements for FAGAs

Future research may take into account the following issues in order to produce effective FAGAs.

Relevant Inputs for the FLCs

Research on determining relevant input variables for the FLCs controlling GA behaviour should be studied in greater depth. These variables should describe either states of the population or features of the chromosomes, so that control parameters may be adapted on the basis thereof to introduce real performance improvements. In this vein, Boulif and Karim [18] claimed recently that previous researches on

FAGAs allowed building FAGA systems that outperform significantly conventional GAs. However, these works, albeit interesting, do not consider the causes that propel the GA in its search for good solutions but rather their effects. Indeed, the fuzzy models use as input either convergence speed, the population diversity or its average fitness (see Table 1). These authors think that it will be more interesting to deal with *cause* inputs first, admitting that nothing forbids complementing them by *effect* inputs. This perspective may prove useful for detecting relevant inputs and determining how to exploit them to adequately tune GA parameters.

Adaptation by Coevolution with Fuzzy Behaviors

The adaptation of GA parameters by coevolution with fuzzy behaviours (FBs) becomes a prospective way for future FAGA works, mainly for two reasons: 1) The use of online learning techniques to derive rule-bases for FAGA has been little explored (see Table 1) and 2) Adaptation of EAs by means of the coevolutionary model is, nowadays, a topic of high interest [171].

Different types of parameter settings were associated with genetic operators, which could be adapted by means of coevolution with FBs. These include the following:

- *Operator probabilities*. There is a type of GA that does not apply both crossover and mutation to the selected solutions. Instead, a set of operators is available, each with a probability of being used, and only one of these is selected to produce offspring. Many adaptive GAs have been designed starting from this GA approach, which adjust the operator probabilities throughout the run [185].
- *Operator parameters*. These parameters determine the way in which genetic operators work. Examples include: 1) the step size of mutation operators for real-coded GAs, which determines the strength in which real genes are mutated, 2) parameters associated with crossover operators for real-coded GAs (see [84]) and dynamic FCB-crossovers [81], 3) the number of parents involved in multi-parent recombination operators, and 4) parameters associated with crossover operators for binary-coded GAs, such as *n*-point crossover and uniform crossover.

  The adaptation at individual-level of operator probabilities and operator parameters by coevolution with FBs may be carried out by considering these variables as a consequence of the fuzzy rules represented in the FBs. Furthermore, appropriate features of the parents should be chosen, as a basis oo which the adjustment of these variable is expressed. On the other hand, hybrid models may be built, in such a way that FBs include information for both the adaptation of operator probabilities and operator parameters. In this case, the model shall detect the operators that should be applied more frequently, along with favourable operator parameter values for them.
- *Mate selection parameters*. In mate selection mechanisms [158], chromosomes carry out the choice of mates for crossover on the basis of their own preferences (which are formulated in terms of different chromosome characteristics, such as the phenotypical distance between individuals).

> Mate selection strategies may be expressed by means of FBs. In particular, given two chromosomes, an FB may induce a probability of mating depending on their characteristics. This probability determines whether or not they are crossed. Then, the process of coevolution with FBs shall discover FBs containing mate selection strategies that encourage recombination between chromosomes that have useful information (characteristics) to exchange.

The adaptive mechanism by coevolution with FBs may also be used for problems where we intuit that particular features of the parents may be taken into account to allow the crossover operator behaviour to be more effective, but we do not know the precise fuzzy rules determining the relation between these features and the appropriate control actions for the operator. In this fashion, this approach allows particular knowledge about the problem to be integrated in the EA in order to improve its behaviour.

4.4.2 Applications and Extensions of Fuzzy EAs

Fuzzy EAs may be defined to tackle particular problems such as multimodal optimisation problems. In addition, fuzzy logic may help modern hybrid metaheuristics to improve their behaviour, obtaining fuzzy hybrid metaheuristics.

Multimodal Optimisation Problems

Given a problem with multiple solutions, a simple EA will tend to converge to a single solution. As a result, various mechanisms have been proposed to stably maintain a diverse population throughout the search, thereby allowing EAs to identify multiple optima reliably. Many of these methods work by encouraging artificial niche formation through sharing and crowding [169], but these methods introduce one or more parameters that affect algorithm performance, parameters such as the sharing radius in fitness sharing or the crowding factor in crowding. In many problems, the uniform specification of niche size is inadequate to capture solutions of varying location and extent without also increasing the population size beyond reasonable bounds. Therefore, there remains a need to develop niching methods that stably and economically find the best niches, regardless of their spacing and extent. FLCs may be useful for the adaptation of parameters associated with sharing and crowding methods. Possible inputs may be: diversity measures, the number of niches that are currently in the population, etc.

Application of Fuzzy Tools to Improve Hybrid Metaheuristics

Over the last years, a large number of search algorithms were reported that do not purely follow the concepts of one single classical metaheuristic, but they attempt to obtain the best from a set of metaheuristics (and even other kinds of optimization methods) that perform together and complement each other to produce a profitable synergy from their combination. These approaches are commonly referred to as hybrid metaheuristics [178]. Memetic algorithms (MAs) [112] are well-known instances of this class of algorithms. They combine an EA in charge of the global search with a local search (LS) procedure, which is executed within the EA run, looking for a synergy that takes benefits from both. The classic scheme of MAs applies the local search procedure on the solutions obtained by the EA with the aim of improving the accuracy of the population members. An important aspect in MAs is the number of fitness function evaluations required by the LS algorithm during their operation (LS intensity). It is fundamental to identify a proper intensity for the LS, because a LS that is too short may be unsuccessful at exploring the neighbourhood of the solution and therefore unsuccessful at improving the search quality. On the other hand, too long LS may backfire by consuming additional fitness evaluations unnecessarily.

A great part of the experience acquired about the application of fuzzy logic to improve EAs may be reused to enhance the behaviour of these innovative search algorithms. For example, FLCs may be designed with the aim of coordinating the different components in a hybrid metaheuristic, assigning different fitness function evaluations to them depending on their specific exploration and/or exploitation features. In particular, for the case of MAs, the adaptation of the LS intensity by FLCs becomes a prospective line of research for obtaining effective MAs.

5 Concluding Remarks

In this chapter, we painted a complete picture of GFSs and fuzzy EAs. In particular, we overviewed important design principles for these algorithms, cited existing literature whenever relevant, provided a taxonomy for each one of them, and discussed future directions and some challenges for these two lines of research. Mainly, this work reveals that GFSs and fuzzy EAs have consolidated backgrounds of knowledge, and therefore, they are two outstanding examples of positive collaboration between soft computing technologies. In addition, it shows that there still remain many exciting research issues connected with these two topics.

References

1. Ah King, R.T.F., Radha, B., Rughooputh, H.C.S.: A fuzzy logic controlled genetic algorithm for optimal electrical distribution network reconfiguration. In: Proc of the 2004 IEEE International Conference on Networking, Sensing and Control, pp. 577–582 (2004)
2. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation 6, 443–462 (2002)
3. Alba, E., Dorronsoro, B.: The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. IEEE Transactions on Evolutionary Computation 9(2), 126–142 (2005)
4. Alcalá, R., Casillas, J., Cordón, O., Herrera, F.: Building fuzzy graphs: features and taxonomy of learning non-grid-oriented fuzzy rule-based systems. International Journal of Intelligent Fuzzy Systems 11, 99–119 (2001)

5. Alcalá, R., Alcalá-Fdez, J., Herrera, F.: A proposal for the genetic lateral tuning of linguistic fuzzy systems and its interaction with rule selection. IEEE Transactions on Fuzzy Systems 15(4), 616–635 (2007)
6. Alcalá, R., Alcalá-Fdez, R., Herrera, F., Otero, J.: Genetic learning of accurate and compact fuzzy rule based systems based on the 2-Tuples linguistic representation. International Journal of Approximate Reasoning 44, 45–64 (2007)
7. Alcalá, R., Gacto, M.J., Herrera, F., Alcalá-Fdez, J.: A multi-objective genetic algorithm for tuning and rule selection to obtain accurate and compact linguistic fuzzy rule-based systems. International Journal of Uncertainty. Fuzziness and Knowledge-Based Systems 15(5), 521–537 (2007)
8. Alcalá-Fdez, J., Herrera, F., Marquez, F., Peregrin, A.: Increasing fuzzy rules cooperation based on evolutionary adaptive inference systems. International Journal of Intelligent Systems 22(9), 1035–1064 (2007)
9. Alcalá-Fdez, J., Sánchez, L., García, S., del Jesús, M.J., Ventura, S., Garrell, J.M., Otero, J., Romero, C., Bacardit, J., Rivas, V.M., Fernández, J.C., Herrera, F.: KEEL: A software tool to assess evolutionary algorithms for data mining problems. In: Soft Computing (in press)
10. Arnone, S., Dell'Orto, M., Tettamanzi, A.: Toward a fuzzy government of genetic populations. In: Proc. of the 6th IEEE Conference on Tools with Artificial Intelligence, pp. 585–591. IEEE Computer Society Press, Los Alamitos (1994)
11. Au, W.-H., Chan, K.C.C., Wong, A.K.C.: A fuzzy approach to partitioning continuous attributes for classification. IEEE Transactions on Knowledge and Data Engineering 18(5), 715–719 (2006)
12. Berlanga, F.J., del Jesus, M.J., González, P., Herrera, F., Mesonero, M.: Multiobjective evolutionary induction of subgroup discovery fuzzy rules: A case study in marketing. In: Perner, P. (ed.) ICDM 2006. LNCS, vol. 4065, pp. 337–349. Springer, Heidelberg (2006)
13. Bergmann, A., Burgard, W., Hemker, A.: Adjusting parameters of genetic algorithms by fuzzy control rules. In: Becks, K.-H., Perret-Gallix, D. (eds.) New Computing Techniques in Physics Research III, pp. 235–240. World Scientific Press, Singapore (1994)
14. Bernadó-Mansilla, E., Garrell-Guiu, J.M.: Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. Evolutionary Computation 11(3), 209–238 (2003)
15. Botta, A., Lazzerini, B., Marcelloni, F.: Context adaptation of Mamdani fuzzy systems through new operators tuned by a genetic algorithm. In: Proceedings of the 2006 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2006), Vancouver, Canada, pp. 7832–7839 (2006)
16. Botta, A., Lazzerini, B., Marcelloni, F., Stefanescu, D.C.: Exploiting fuzzy ordering relations to preserve interpretability in context adaptation of fuzzy systems. In: Proceedings of the 2007 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007), London, UK, pp. 1137–1142 (2007)
17. Botta, A., Lazzerini, B., Marcelloni, F., Stefanescu, D.C.: Context Adaptation of Fuzzy Systems Through a Multi-objective Evolutionary Approach Based on a Novel Interpretability Index. Soft Computing 13(3), 437–449 (2009)
18. Boulif, M., Atif, K.: A new fuzzy genetic algorithm for the dynamic bi-objective cell formation problem considering passive and active strategies. International Journal of Approximate Reasoning 47, 141–165 (2008)
19. Cano, J.R., Herrera, F., Lozano, M.: Evolutionary stratified training set selection for extracting classification rules with trade-off precision-interpretability. Data and Knowledge Engineering 60, 90–108 (2007)

20. Cantú-Paz, E.: Efficient and accurate parallel genetic algorithms. Book Series on Genetic Algorithms and Evolutionary Computation. Kluwer, Norwell (2000)
21. Carse, B., Fogarty, T.C., Munro, A.: Evolving fuzzy rule based controllers using genetic algorithms. Fuzzy Sets and Systems 80(3), 273–293 (1996)
22. Casillas, J., Carse, B., Bull, L.: Fuzzy-XCS: A Michigan genetic fuzzy system. IEEE Transactions on Fuzzy Systems 15(4), 536–550 (2007)
23. Casillas, J., Cordón, O., Herrera, F., del Jesus, M.J.: Genetic feature selection in a fuzzy rule-based classification system learning process for high-dimensional problems. Information Sciences 136(1-4), 135–157 (2001)
24. Casillas, J., Cordón, O., del Jesus, M.J., Herrera, F.: Genetic tuning of fuzzy rule deep structures preserving interpretability for linguistic modeling. IEEE Trans. on Fuzzy Systems 13(1), 13–29 (2005)
25. Casillas, J., Cordón, O., Herrera, F., Magdalena, L. (eds.): Accuracy improvements in linguistic fuzzy modeling. Springer, Berlin (2003)
26. Casillas, J., Cordón, O., Herrera, F., Magdalena, L. (eds.): Interpretability issues in fuzzy modeling. Springer, Berlin (2003)
27. Casillas, J., Martínez, P.: Consistent, complete and compact generation of DNF-type fuzzy rules by a Pittsburgh-style genetic algorithm. In: Proceedings of the 2007 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007), London, UK, pp. 1745–1750 (2007)
28. Casillas, J., Martínez, P., Benítez, A.D.: Learning consistent, complete and compact fuzzy rules sets in conjunctive normal form for system identification. Soft Computing 13(3), 451–465 (2009)
29. Chen, C.-H., Hong, T.-P., Tseng, V.S., Lee, C.-S.: A genetic-fuzzy mining approach for items with multiple minimum supports. Soft Computing 13(3), 521–533 (2009)
30. Cherkassky, V., Mulier, F.: Learning from data: concepts, theory and methods. John Wiley & Sons, New York (1998)
31. Mc Clintock, S., Lunney, T., Hashim, A.: Using fuzzy logic to optimize genetic algorithm performance. In: Proceedings of 1997 IEEE International Conference on Intelligent Engineering Systems, Budapest, Hungary, pp. 271–275 (1997)
32. Mc Clintock, S., Lunney, T., Hashim, A.: A fuzzy logic controlled genetic algorithm environment. In: Proceedings of 1997 IEEE International Conference on Systems, Man, and Cybernetics, Orlando, Florida, USA, pp. 2181–2186 (1997)
33. Cococcioni, M., Ducange, P., Lazzerini, B., Marcelloni, F.: A Pareto-based multi-objective evolutionary approach to the identification of Mamdani fuzzy systems. Soft Computing 11(11), 1013–1031 (2007)
34. Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary algorithms for solving multi-objective problems. Kluwer Academic Publishers, Dordrecht (2002)
35. Cordón, O., del Jesús, M.J., Herrera, F., Lozano, M.: MOGUL: A Methodology to Obtain Genetic fuzzy rule-based systems Under the iterative rule Learning approach. International Journal of Intelligent Systems 14, 1123–1153 (1999)
36. Cordón, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L.: Ten years of genetic fuzzy systems: current framework and new trends. Fuzzy Sets and Systems 141, 5–31 (2004)
37. Cordón, O., Herrera, F.: A three-stage evolutionary process for learning descriptive and approximate fuzzy-logic-controller knowledge bases from examples. International Journal of Approximate Reasoning 17(4), 369–407 (1997)
38. Cordón, O., Herrera, F., Hoffmann, F., Magdalena, L.: Genetic fuzzy systems. In: Evolutionary tuning and learning of fuzzy knowledge bases. World Scientific, Singapore (2001)

39. Cordón, O., Herrera, F., Villar, P.: Analysis and guidelines to obtain a good fuzzy partition granularity for fuzzy rule-based systems using simulated annealing. International Journal of Approximate Reasoning 25(3), 187–215 (2000)
40. Cordón, O., Herrera, F., Magdalena, L., Villar, P.: A genetic learning process for the scaling factors, granularity and contexts of the fuzzy rule-based system data base. Information Sciences 136, 85–107 (2001)
41. Cordón, O., Herrera, F., Villar, P.: Generating the knowledge base of a fuzzy rule-based system by the genetic learning of data base. IEEE Transactions on Fuzzy Systems 9(4), 667–674 (2001)
42. Crockett, K.A., Bandar, Z., Fowdar, J., O'Shea, J.: Genetic tuning of fuzzy inference within fuzzy classifier systems. Expert Systems with Applications 23, 63–82 (2006)
43. Crockett, K., Bandar, Z., Mclean, D.: On the optimization of T-norm parameters within fuzzy decision trees. In: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007), London, UK, pp. 103–108 (2007)
44. Das, D.: Optimal placement of capacitors in radial distribution system using a Fuzzy-GA method. International Journal of Electrical Power & Energy Systems (in press)
45. Deb, K.: Multi-objective optimization using evolutionary algorithms. John Wiley & Sons, Chichester (2001)
46. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6(2), 182–197 (2002)
47. De Jong, K.A., Spears, W.M., Gordon, D.F.: Using genetic algorithms for concept learning. Machine Learning 13, 161–188 (1993)
48. del Jesus, M.J., González, P., Herrera, F., Mesonero, M.: Evolutionary fuzzy rule induction process for subgroup discovery: A case study in marketing. IEEE Transactions on Fuzzy Systems 15(4), 578–592 (2007)
49. Demsar, J.: Statistical comparison of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
50. Diettereich, T.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation 10, 1895–1924 (1998)
51. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press, Cambridge (2004)
52. Dozier, G.V., McCullough, S., Homaifar, A., Moore, L.: Multiobjective evolutionary path planning via fuzzy tournament selection. In: IEEE International Conference on Evolutionary Computation (ICEC 1998), pp. 684–689. IEEE Press, Piscataway (1998)
53. Driankow, D., Hellendoorn, H., Reinfrank, M.: An introduction to fuzzy control. Springer, Berlin (1993)
54. Dubois, D., Prade, H., Sudkamp, T.: On the representation, measurement, and discovery of fuzzy associations. IEEE Trans. on Fuzzy Systems 13, 250–262 (2005)
55. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. Evolutionary Computation 3(2), 124–141 (1999)
56. Eiben, A.E., Smith, J.E.: Introduction to evolutionary computation. Springer, Berlin (2003)
57. Fayyad, U., Piatesky-Shapiro, G., Smyth, P.: From data mining from knowledge discovery in databases. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery & Data Mining, pp. 1–34. AAAI/MIT (1996)
58. Fogel, D.B.: An introduction to simulated evolutionary optimization. IEEE Transactions on Neural Networks 5(1), 3–14 (1994)
59. Freitas, A.A.: Data mining and knowledge discovery with evolutionary algorithms. Springer, Berlin (2002)
60. Gacto, M.J., Alcalá, R., Herrera, F.: Adaptation and application of multi-objective evolutionary algorithms for rule reduction and parameter tuning of fuzzy rule-based systems. Soft Computing 13(3), 419–436 (2009)

61. Geyer-Schulz, A.: Fuzzy rule-based expert systems and genetic machine learning. Physica-Verlag, Berlin (1995)
62. Giordana, A., Neri, F.: Search-intensive concept induction. Evolutionary Computation 3, 375–416 (1995)
63. Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading (1989)
64. González, A., Pérez, R.: SLAVE: A genetic learning system based on an iterative approach. IEEE Transactions on Fuzzy Systems 27, 176–191 (1999)
65. González, A., Pérez, R.: Selection of relevant features in a fuzzy genetic learning algorithm. IEEE Transactions on Systems, Man and Cybernetics. Part B: Cybernetics 31(3), 417–425 (2001)
66. González, A., Pérez, R.: An analysis of the scalability of an embedded feature selection model for classification problems. In: Proc. Eleventh Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU 2006), Paris, pp. 1949–1956 (2006)
67. Greene, D.P., Smith, S.F.: Competition-based induction of decision models from examples. Machine Learning 3, 229–257 (1993)
68. Grefenstette, J.J.: Optimization of control parameters for genetic algorithms. IEEE Trans Systems, Man, and Cybernetics 16, 122–128 (1986)
69. Gudwin, R.R., Gomide, F.A.C., Pedrycz, W.: Context adaptation in fuzzy processing and genetic algorithms. International Journal of Intelligent Systems 13(10-11), 929–948 (1998)
70. Hamzeh, A., Rahmani, A., Parsa, N.: Intelligent exploration method to adapt exploration rate in XCS, based on adaptive fuzzy genetic algorithm. In: Proc. of the 2006 IEEE Conference on Cybernetics and Intelligent Systems, pp. 1–6 (2006)
71. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Mining & Knowledge Discovery 15(1), 55–86 (2007)
72. Herrera, F.: Genetic fuzzy systems: Status, critical considerations and future directions. International Journal of Computational Intelligence Research 1(1), 59–67 (2005)
73. Herrera, F.: Genetic fuzzy systems: taxonomy, current research trends and prospects. Evolutionary Intelligence 1, 27–46 (2008)
74. Herrera, F., Lozano, M.: Adaptation of genetic algorithm parameters based on fuzzy logic controllers. In: Herrera, F., Verdegay, J.L. (eds.) Genetic Algorithms and Soft Computing, pp. 95–125. Physica-Verlag (1996)
75. Herrera, F., Lozano, M.: Heuristic crossover for real-coded genetic algorithms based on fuzzy connectives. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 336–345. Springer, Heidelberg (1996)
76. Herrera, F., Lozano, M.: Adaptive control of the mutation probability by fuzzy logic controllers. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 335–344. Springer, Heidelberg (2000)
77. Herrera, F., Lozano, M.: Adaptive genetic operators based on coevolution with fuzzy behaviours. IEEE Trans. on Evolut. Comput. 5(2), 1–18 (2001)
78. Herrera, F., Lozano, M.: Fuzzy adaptive genetic algorithms: design, taxonomy, and future directions. Soft Computing 7, 545–562 (2003)
79. Herrera, F., Lozano, M., Verdegay, J.L.: Tackling fuzzy genetic algorithms. In: Genetic Algorithms in Engineering and Computer Science, pp. 167–189. John Wiley, New York (1995)
80. Herrera, F., Lozano, M., Verdegay, J.L.: Tuning fuzzy-logic controllers by genetic algorithms. International Journal of Approximate Reasoning 12(3-4), 299–315 (1995)

81. Herrera, F., Lozano, M., Verdegay, J.L.: Dynamic and heuristic fuzzy connectives-based crossover operators for controlling the diversity and convengence of real-coded genetic algorithms. Int. Journal of Intelligent Systems 11, 1013–1041 (1996)
82. Herrera, F., Lozano, M., Verdegay, J.L.: Fuzzy connectives based crossover operators to model genetic algorithms population diversity. Fuzzy Sets and Systems 92(1), 21–30 (1997)
83. Herrera, F., Lozano, M., Verdegay, J.L.: A learning process for fuzzy control rules using genetic algorithms. Fuzzy Sets and Systems 100, 143–151 (1998)
84. Herrera, F., Lozano, M., Sánchez, A.M.: A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study. International Journal of Intelligent Systems 18, 309–338 (2003)
85. Homaifar, A., Mccormick, E.: Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. IEEE Transactions on Fuzzy Systems 3(2), 129–139 (1995)
86. Hoffmann, F., Schauten, D., Hölemann, S.: Incremental evolutionary design of TSK fuzzy controllers. IEEE Transactions on Fuzzy Systems 15(4), 563–577 (2007)
87. Holland, J.H.: Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor (1975)
88. Holland, J.H., Reitman, J.S.: Cognitive systems based on adaptive algorithms. In: Waterman, D.A., Hayes-Roth, F. (eds.) Patter-Directed Inference Systems. Academic Press, London (1978)
89. Hong, T.P., Chen, C.H., Wu, Y.L., et al.: A GA-based fuzzy mining approach to achieve a trade-off between number of rules and suitability of membership functions. Soft Computing 10(11), 1091–1101 (2006)
90. Hüllermeier, E.: Fuzzy methods in machine learning and data mining: Status and prospects. Fuzzy Sets and Systems 156(3), 387–406 (2005)
91. Ishibuchi, H.: Multiobjective genetic fuzzy systems: review and future research directions. In: Proceedings of the 2007 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007), London, UK, pp. 913–918 (2007)
92. Ishibuchi, H., Murata, T., Turksen, I.B.: Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. Fuzzy Sets and Systems 8(2), 135–150 (1997)
93. Ishibuchi, H., Nakashima, T., Murata, T.: Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. IEEE Transactions on Systems, Man and Cybernetics. Part B-Cybernetics 29(5), 601–618 (1999)
94. Ishibuchi, H., Nakashima, T., Nii, M.: Classification and modeling with linguistic information granules: Advanced approaches to linguistic data mining. Springer, Berlin (2004)
95. Ishibuchi, H., Nozaki, K., Yamamoto, N., Tanaka, H.: Selection fuzzy IF-THEN rules for classification problems using genetic algorithms. IEEE Transactions on Fuzzy Systems 3(3), 260–270 (1995)
96. Ishibuchi, H., Yamamoto, T.: Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. Fuzzy Sets and Systems 141(1), 59–88 (2004)
97. Juang, C.F., Lin, J.Y., Lin, C.T.: Genetic reinforcement learning through symbiotic evolution for fuzzy controller design. IEEE Transactions on Systems, Man and Cybernetics. Part B-Cybernetics 30(2), 290–302 (2000)
98. Kacem, I., Hammadi, S., Borne, P.: Pareto-optimality approach based on uniform design and fuzzy evolutionary algorithms for flexible job-shop scheduling problems (FJSPs). In: 2002 IEEE International Conference on Systems, Man and Cybernetics, p. 7 (2002)

99. Kang, Q., Wang, L., Wu, Q.: Research on fuzzy adaptive optimization strategy of particle swarm algorithm. International Journal of Information Technology 12(3), 65–77 (2006)
100. Karr, C.: Genetic algorithms for fuzzy controllers. AI Expert 6(2), 26–33 (1991)
101. Kaya, M.: Multi-objective genetic algorithm based approaches for mining optimized fuzzy association rules. Soft Computing 10(7), 578–586 (2006)
102. Kaya, M., Alhajj, R.: Genetic algorithm based framework for mining fuzzy association rules. Fuzzy Sets and Systems 152(3), 587–601 (2005)
103. Kennedy, J., Eberhart, R.C.: Swarm intelligence. Morgan Kauffmann, San Francisco (2001)
104. Kiliç, S., Kahraman, C.: Metaheuristic techniques for job shop scheduling problem and a fuzzy ant colony optimization algorithm. Studies in Fuzziness and Soft Computing 201, 401–425 (2006)
105. Kim, D., Choi, Y., Lee, S.: An accurate COG defuzzifier design using Lamarckian coadaptation of learning and evolution. Fuzzy Sets Syst. 130(2), 207–225 (2002)
106. King, R.T.F.A., Radha, B., Rughooputh, H.C.S.: A fuzzy logic controlled genetic algorithm for optimal electrical distribution network reconfiguration. In: Proc. of 2004 IEEE International Conference on Networking, Sensing and Control, Taipei, Taiwan, pp. 577–582 (2004)
107. Klir, G., Yuan, B.: Fuzzy sets and fuzzy logic; theory and applications. Prentice-Hall, Englewood Cliffs (1995)
108. Klösgen, W.: EXPLORA: a multipattern and multistrategy discovery assistant. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 249–271. MIT Press, Cambridge (1996)
109. Konar, A.: Computational intelligence: principles, techniques and applications. Springer, Berlin (2005)
110. Kovacs, T.: Strength or accuracy: credit assignment in learning classifier systems. Springer, Berlin (2004)
111. Koza, J.R.: Genetic programing: on the programming of computers by means of natural selection. The MIT Press, Cambridge (1992)
112. Krasnogor, N., Smith, J.E.: A tutorial for competent memetic algorithms: model, taxonomy, and design issue. IEEE Trans. Evol. Comput. 9(5), 474–488 (2005)
113. Kuncheva, L.: Fuzzy classifier design. Springer, Berlin (2000)
114. Kweku-Muata, Osey-Bryson: Evaluation of decision trees: a multicriteria approach. Computers and Operations Research 31, 1933–1945 (2004)
115. Last, M., Eyal, S.: A fuzzy-based lifetime extension of genetic algorithms. Fuzzy Sets and Systems 149, 131–147 (2005)
116. Last, M., Eyal, S., Kandel, A.: Effective black-box testing with genetic algorithms. In: Ur, S., Bin, E., Wolfsthal, Y. (eds.) HVC 2005. LNCS, vol. 3875, pp. 134–148. Springer, Heidelberg (2006)
117. Lau, H.C.W., Chan, T.M., Tsui, W.T.: Fuzzy logic guided genetic algorithms for the location assignment of items. In: 2007 IEEE Congress on Evolutionary Computation (CEC 2007), pp. 4281–4288 (2007)
118. Lavra, N., Cestnik, B., Gamberger, D., Flach, P.: Decision support through subgroup discovery: three case studies and the lessons learned. Machine Learning 57, 115–143 (2004)
119. Lee, M.A., Esbensen, H.: Fuzzy/multiobjective genetic systems for intelligent systems design tools and components. In: Fuzzy Evolutionary Computation, pp. 57–80. Kluwer Academic Publishers, Dordrecht (1997)

120. Lee, M.A., Takagi, H.: Dynamic control of genetic algorithms using fuzzy logic techniques. In: Proc of the Fifth Int Conf on Genetic Algorithms, pp. 76–83. Morgan Kaufmann, San Francisco (1993)

121. Lee, M.A., Takagi, H.: A framework for studying the effects of dynamic crossover, mutation, and population sizing in genetic algorithms. In: Advances in Fuzzy Logic, Neural Networks and Genetic Algorithms. LNCS, vol. 1011, pp. 111–126. Springer, Heidelberg (1994)

122. Li, Q., Tong, X., Xie, S., Liu, G.: An improved adaptive algorithm for controlling the probabilities of crossover and mutation based on a fuzzy control strategy. In: Proc. of the 6th International Conference on Hybrid Intelligent Systems and 4th Conference on Neuro-Computing and Evolving Intelligence, p. 50. IEEE Computer Society, Los Alamitos (2006)

123. Li, Q., Yin, Y., Wang, Z., Liu, G.: Comparative studies of fuzzy genetic algorithms. In: Liu, D., Fei, S., Hou, Z., Zhang, H., Sun, C. (eds.) ISNN 2007. LNCS, vol. 4492, pp. 251–256. Springer, Heidelberg (2007)

124. Hongbo Liu, H., Abraham, A.: A Fuzzy adaptive turbulent particle swarm optimization. In: Proc. Fifth International Conference on Hybrid Intelligent Systems, pp. 445–450 (2005)

125. Liu, H., Abraham, A.: An hybrid fuzzy variable neighborhood particle swarm optimization algorithm for solving quadratic assignment problems. Journal of Universal Computer Science 13(9), 1309–1331 (2007)

126. Liu, H., Xu, Z., Abraham, A.: Hybrid fuzzy-genetic algorithm approach for crew grouping. In: Proceedings of the 2005, 5th International Conference on Intelligent Systems Design and Applications (ISDA 2005), pp. 332–337 (2005)

127. Liu, J., Lampinen, J.: Adaptive parameter control of differential evolution. In: Proceedings of the 8th International Mendel Conference on soft computing, pp. 19–26 (2002)

128. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. In: Proceedings of the 17th IEEE region 10th International Conference on computer, communications, control and power engineering, pp. 606–611 (2002)

129. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft Comput. 9, 448–462 (2005)

130. Maeda, Y.: Fuzzy adaptive search method for genetic programming. International Journal of Advanced Computational Intelligence 3(2), 131–135 (1999)

131. Maeda, Y., Ishita, M., Li, Q.: Fuzzy adaptive search method for parallel genetic algorithm with island combination process. International Journal of Approximate Reasoning 41, 59–73 (2006)

132. Maeda, Y., Li, Q.: Fuzzy adaptive search method for parallel genetic algorithm tuned by evolution degree based on diversity measure. In: Melin, P., Castillo, O., Aguilar, L.T., Kacprzyk, J., Pedrycz, W. (eds.) IFSA 2007. LNCS (LNAI), vol. 4529, pp. 677–687. Springer, Heidelberg (2007)

133. Magdalena, L.: Adapting the gain of an FLC with genetic algorithms. International Journal of Approximate Reasoning 17(4), 327–349 (1997)

134. Mamdani, E.H.: Applications of fuzzy algorithm for control a simple dynamic plant. Proceedings of the IEEE 121(12), 1585–1588 (1974)

135. Márquez, F.A., Peregrín, A., Herrera, F.: Cooperative evolutionary learning of linguistic fuzzy rules and parametric aggregation connectors for Mamdani fuzzy systems. IEEE Trans. on Fuzzy Systems 15(6), 1162–1178 (2008)

136. Matousek, R., Osmera, P., Roupec, J.: GA with fuzzy inference system. In: Proceedings of the 2000 Congress on Evolutionary Computation, pp. 646–651 (2000)

137. Meyer, L., Feng, X.: A fuzzy stop criterion for genetic algorithms using performance estimation. In: Proc. Third IEEE Int. Conf. on Fuzzy Systems, pp. 1990–1995 (1994)
138. Mikut, R., Jäkel, J., Gröll, L.: Interpretability issues in data-based learning of fuzzy systems. Fuzzy Sets and Systems 150, 179–197 (2005)
139. Moriarty, D.E., Miikkulainen, R.: Efficient reinforcement learning through symbiotic evolution. Machine Learning 22, 11–32 (1996)
140. Mucientes, M., Vidal, J.C., Bugarín, A., Lama, M.: Processing time estimations by variable structure TSK rules learned through genetic programming. Soft Computing 13(3), 497–509 (2009)
141. Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive models for the breeder genetic algorithm I. continuous parameter optimization. Evolutionary Computation 1, 25–49 (1993)
142. Mirabedini, S.J., Teshnehlab, M.: Performance evaluation of fuzzy ant based routing method for connectionless networks. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007. LNCS, vol. 4488, pp. 960–965. Springer, Heidelberg (2007)
143. Nojima, Y., Kuwajima, I., Ishibuchi, H.: Data set subdivision for parallel distribution implementation of genetic fuzzy rule selection. In: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007), London, UK, pp. 2006–2011 (2007)
144. Nojima, Y., Ishibuchi, H., Kuwajima, I.: Parallel distributed genetic fuzzy rule selection. Soft Computing 13(3), 511–519 (2009)
145. Orriols-Puig, A., Casillas, J., Bernadó-Mansilla, E.: Fuzzy-UCS: preliminary results. In: 10th International Workshop on Learning Classifier Systems (IWLCS 2007), London, UK, pp. 2871–2874 (2007)
146. Palm, R., Driankov, D.: Model based fuzzy control. Springer, Berlin (1997)
147. Park, D., Kandel, A., Langholz, G.: Genetic-based new fuzzy-reasoning models with applications to fuzzy control. IEEE Transactions on Systems, Man and Cybernetics 24(1), 39–47 (1994)
148. Pedrycz, W. (ed.): Fuzzy modelling: paradigms and practice. Kluwer Academic Press, Dordrecht (1996)
149. Pedrycz, W.: Fuzzy evolutionary computing. Soft Computing 2, 61–72 (1998)
150. Pham, D.T., Karaboga, D.: Optimum design of fuzzy logic controllers using genetic algorithms. Journal of Systems Engineering 1, 114–118 (1991)
151. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential evolution: a practical approach to global optimization. Springer, Heidelberg (2005)
152. Rachmawati, L., Srinivasan, D.: A hybrid fuzzy evolutionary algorithm for a multi-objective resource allocation problem. In: Proc. of the Fifth International Conference on Hybrid Intelligent Systems (HIS 2005), pp. 55–60 (2005)
153. Regattieri-Delgado, M., Yassue-Nagai, E., Ramos de Arruda, L.V.: A neuro-coevolutionary GFS to build soft sensors. Soft Computing 13(3), 481–495 (2009)
154. Reynolds, R.G.: An introduction to cultural algorithms. In: Proc. of the 3rd Annual Conference on Evolutionary Programming, pp. 131–139. World Scientific, Singapore (1994)
155. Reynolds, R.G., Chung, C.J.: Regulating the amount of information used for self-adaptation in cultural algorithms. In: Proc. of the Seventh Int. Conf. on Genetic Algorithms, pp. 401–408. Morgan Kaufmann Publishers, San Francisco (1997)
156. Richter, J.N.: Fuzzy evolutionary cellular automata. Master thesis, UT State University, Logan, Utah (2003)
157. Rojas, R.: Neural networks: a systematic introduction. Springer, Berlin (1996)
158. Ronald, E.: When selection meets seduction. In: Proc of the Fifth Int Conf on Genetic Algorithms, pp. 167–173. Morgan Kaufmann, San Francisco (1993)

159. Sahoo, N.C., Ranjan, R., Prasad, K., Chaturvedi, A.: A fuzzy-tuned genetic algorithm for optimal reconfigurations of radial distribution network. European Trans. Electr. Power 17, 97–111 (2006)
160. Sahoo, N.C., Prasad, K.: A fuzzy genetic approach for network reconfiguration to enhance voltage stability in radial distribution systems. Energy Conversion and Management 47, 3288–3306 (2006)
161. Sánchez, L., Casillas, J., Cordón, O., del Jesus, M.J.: Some relationships between fuzzy and random classifiers and models. International Journal of Approximate Reasoning 29, 175–213 (2001)
162. Sánchez, L., Couso, I.: Advocating the use of imprecisely observed data in genetic fuzzy systems. IEEE Transactions on Fuzzy Systems 15(4), 551–562 (2007)
163. Sánchez, L., Otero, J., Couso, I.: Obtaining Linguistic Fuzzy Rule-based Regression Models from Imprecise Data with Multiobjective Genetic Algorithms. Soft Computing 13(3), 467–479 (2009)
164. Sebban, M., Nock, R., Cahuchat, J.H., Rakotomalala, R.: Impact of learning set quality and size on decision tree performance. Int. J. of Computers, Syst. and Signals 1, 85–105 (2000)
165. Setnes, M., Roubos, H.: GA-fuzzy modeling and classification: complexity and performance. IEEE Transactions on Fuzzy Systems 8(5), 509–522 (2000)
166. Setzkorn, C., Paton, R.C.: On the use of multi-objective evolutionary algorithms for the induction of fuzzy classification rule systems. BioSystems 81, 101–112 (2005)
167. Sharma, S.K., Irwin, G.W.: Fuzzy coding of genetic algorithms. IEEE Trans. on Evolutionary Computation 7(4), 344–355 (2003)
168. Shi, Y., Eberhart, R., Chen, Y.: Implementation of evolutionary fuzzy systems. IEEE Trans Fuzzy Systems 7(2), 109–119 (1999)
169. Gulshan, S., Kalyanmoy, D.: Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: Proc. of the 8th annual conference on Genetic and evolutionary computation, pp. 1305–1312 (2006)
170. Smith, S.: A learning system based on genetic algorithms. Ph.D. thesis. Unversity of Pittsburgh (1980)
171. Smith, J.E.: Coevolving memetic algorithms: a review and progress report. IEEE Transaction on Systems, Man, and Cybernetics Part B: Cybernetics 37(1), 6–17 (2007)
172. Song, Y.H., Wang, G.S., Johns, A.T., Wang, P.Y.: Improved genetic algorithms with fuzzy logic controlled crossover and mutation. In: UKACC International Conference on CONTROL 1996, pp. 140–144 (1996)
173. Song, Y.H., Wang, G.S., Wang, P.T., Johns, A.T.: Environmental/economic dispatch using fuzzy logic controlled genetic algorithms. IEEE Proc. on Generation, Transmission and Distribution 144(4), 377–382 (1997)
174. Streifel, R.J., Marks II, R.J., Reed, R., Choi, J.J., Healy, M.: Dynamic fuzzy control of genetic algorithm parameter coding. IEEE Trans Systems, Man, and Cybernetics - Part B: Cybernetics 29(3), 426–433 (1999)
175. Subbu, R., Sanderson, A.C., Bonissone, P.: Fuzzy logic controlled genetic algorithms versus tuned genetic algorithms: An agile manufacturing application. In: Proc. ISIC/CIRA/ISAS Conf., pp. 434–440 (1998)
176. Subbu, R., Bonissone, P.: A retrospective view of fuzzy control of evolutionary algorithm resources. In: Proc. IEEE Int. Conf. Fuzzy Syst., pp. 143–148 (2003)
177. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its application to modelling and control. IEEE Transactions on Systems, Man and Cybernetics 15(1), 116–132 (1985)
178. Talbi, E.-G.: A taxonomy of hybrid metaheuristics. J. Heuristics 8(5), 541–565 (2002)

179. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Pearson, Boston (2006)
180. Thrift, P.: Fuzzy logic synthesis with genetic algorithms. In: Proc. of 4th International Conference on Genetic Algorithms (ICGA 1991), pp. 509–513 (1991)
181. Teodorovic, D., Lucic, P.: Schedule synchronization in public transit using the fuzzy ant system. Transportation Planning and Technology 28(1), 47–76 (2005)
182. Tettamanzi, A.G.: Evolutionary algorithms and fuzzy logic: a two-way integration. In: 2nd Joint Conference on Information Sciences, pp. 464–467 (1995)
183. Tettamanzi, A., Tomassini, M.: Fuzzy evolutionary algorithms. In: Soft Computing: Integrating Evolutionary, Neural, and Fuzzy Systems, pp. 233–248. Springer, Heidelberg (2001)
184. Tsang, C.-H., Tsai, J.H., Wang, H.: Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. Pattern Recognition 40(9), 2373–2391 (2007)
185. Tuson, A.L., Ross, P.: Adapting operator settings in genetic algorithms. Evolut. Comput. 6(2), 161–184 (1998)
186. Valenzuela-Rendon, M.: The fuzzy classifier system: A classifier system for continuously varying variables. In: Proc. of 4th International Conference on Genetic Algorithms (ICGA 1991), pp. 346–353 (1991)
187. Valenzuela-Rendon, M.: Reinforcement learning in the fuzzy classifier system. Expert Systems with Applications 14, 237–247 (1998)
188. Venturini, G.: SIA: a supervised inductive algorithm with genetic search for learning attribute based concepts. In: Brazdil, P.B. (ed.) ECML 1993. LNCS, vol. 667, pp. 280–296. Springer, Heidelberg (1993)
189. Voget, S.: Multiobjective optimization with genetic algorithms and fuzzy-control. In: Proc of the Fourth European Congress on Intelligent Techniques and Soft Computing, pp. 391–394 (1996)
190. Voget, S., Kolonko, M.: Multidimensional optimization with a fuzzy genetic algorithm. Journal of Heuristic 4(3), 221–244 (1998)
191. Voigt, H.M.: Fuzzy evolutionary algorithms. Technical Report tr-92-038, International Computer Science Institute (ICSI), Berkeley (1992)
192. Voigt, H.M.: Soft genetic operators in evolutionary algorithms. In: Banzhaf, W., Eckman, F.H. (eds.) Evolution as a Computational Process 1992. LNCS, vol. 899, pp. 123–141. Springer, Heidelberg (1995)
193. Voigt, H.M., Anheyer, T.: Modal mutations in evolutionary algorithms. In: Proceeding of the First IEEE International Conference on Evolutionary Computation, pp. 88–92. IEEE Press, Los Alamitos (1994)
194. Voigt, H.M., Born, J., Santibáñez-Koref, I.: A multivalued evolutionary algorithms. Technical Report tr-93-022, International Computer Science Institute (ICSI), Berkeley (1993)
195. Voigt, H.M., Mühlenbein, H., Cvetković, D.: Fuzzy recombination for the breeder genetic algorithm. In: Proc. of the Sixth Int. Conf. on Genetic Algorithms, pp. 104–111. Morgan Kaufmann Publishers, San Francisco (1995)
196. Wang, K.: A new fuzzy genetic algorithms based on population diversity. In: Proc. of 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp. 108–112 (2001)
197. Wang, H., Kwong, S., Jin, Y., Wei, W., Man, K.F.: Multiobjective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction. Fuzzy Sets and Systems 149, 49–186 (2005)

198. Wang, P.Y., Wang, G.S., Song, Y.H., Johns, A.T.: Fuzzy logic controlled genetic algorithms. In: Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, pp. 972–979 (1996)
199. Wilson, S.: Classifier fitness based on accuracy. Evol. Comput. 3(2), 149–175 (1995)
200. Wong, M.L., Leung, K.S.: Data mining using grammar based genetic programming and applications. Kluwer Academic Publishers, Dordrecht (2000)
201. Wrobel, S.: An algorithm for multi-relational discovery of subgroups. In: Komorowski, J., Żytkow, J.M. (eds.) PKDD 1997. LNCS, vol. 1263, pp. 78–87. Springer, Heidelberg (1997)
202. Xu, H.Y., Vukovich, G.: A fuzzy genetic algorithm with effective search and optimization. In: Proc. of 1993 International Joint Conference on Neural Networks, pp. 2967–2970 (1993)
203. Xu, H.Y., Vukovich, G., Ichikawa, Y., Ishii, Y.: Fuzzy evolutionary algorithms and automatic robot trajectory generation. In: Proceeding of the First IEEE International Conference on Evolutionary Computation, pp. 595–600. IEEE Press, Los Alamitos (1994)
204. Yager, R.R., Filev, D.P.: Essentials of fuzzy modeling and control. John Wiley & Sons, Chichester (1994)
205. Yamakawa, T.: Stabilization of an inverted pendulum by a high-speed fuzzy logic controller hardware system. Fuzzy Sets and Systems 32, 161–180 (1989)
206. Yang, Q., Wu, X.: 10 challenging problems in data mining research. International Journal of Information Technology & Decision Making 5(4), 597–604 (2006)
207. Yun, Y., Gen, M.: Performance analysis of adaptive genetic algorithm with fuzzy logic and heuristics. Fuzzy Optimization and Decision Making 2, 161–175 (2003)
208. Zadeh, L.A.: Fuzzy sets. Information and Control 8, 338–353 (1965)
209. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Proc. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems (EUROGEN 2001), Barcelona, Spain, pp. 95–100 (2001)
210. Zeng, X., Rabenasolo, B.: A fuzzy logic based design for adaptive genetic algorithms. In: Proc. of the European Congress on Intelligent Techniques and Soft Computing, pp. 660–664 (1997)
211. Zhu, L., Zhang, H., Jing, Y.: A new neuro-fuzzy adaptive genetic algorithm. Journal of Electronic Science and Technology of China 1(1) (2003)

Multiobjective Genetic Fuzzy Systems

Hisao Ishibuchi and Yusuke Nojima

Abstract. In the design of fuzzy rule-based systems, we have two conflicting goals: One is accuracy maximization, and the other is complexity minimization (i.e., interpretability maximization). There exists a tradeoff relation between these two goals. That is, we cannot simultaneously achieve accuracy maximization and complexity minimization. Various approaches have been proposed to find accurate and interpretable fuzzy rule-based systems. In some approaches, these two goals are integrated into a single objective function which can be optimized by standard single-objective optimization techniques. In other approaches, accuracy maximization and complexity minimization are handled as different objectives in the framework of multiobjective optimization. Recently, multiobjective genetic algorithms have been used to search for a large number of non-dominated fuzzy rule-based systems along the accuracy-complexity tradeoff surface in some studies. These studies are often referred to as multiobjective genetic fuzzy systems. In this chapter, we first briefly explain the concept of accuracy-complexity tradeoff in the design of fuzzy rule-based systems. Next we explain various studies in multiobjective genetic fuzzy systems. Two basic ideas are explained in detail through computational experiments. Then we review a wide range of studies related to multiobjective genetic fuzzy systems. Finally we point out future research directions.

1 Introduction

One advantage of fuzzy rule-based systems over other nonlinear systems such as neural networks is their linguistic interpretability. That is, each fuzzy rule is linguistically interpretable when fuzzy rule-based systems are designed from linguistic knowledge of human experts. Linguistic knowledge, however, is not always

Hisao Ishibuchi and Yusuke Nojima
Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University
1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan
e-mail: hisaoi@cs.osakafu-u.ac.jp,nojima@cs.osakafu-u.ac.jp

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 131–173.
springerlink.com

available. Thus various approaches have been proposed for extracting fuzzy rules from numerical data in the literature (e.g., Takagi and Sugeno [136], and Wang and Mendel [142]). Learning techniques of neural networks (e.g., the back-propagation algorithm [131]) have been applied to fuzzy rule-based systems for their parameter tuning [3, 66, 92, 115]. Fuzzy rule-based systems with learning capability are often called neuro-fuzzy systems [115]. ANFIS of Jang [92] is the most well-known and frequently-used neuro-fuzzy system. Whereas only continuous parameters are adjusted in neuro-fuzzy systems, evolutionary optimization techniques can be used not only for parameter tuning but also for discrete optimization such as input selection, rule generation and rule selection. Genetic algorithms [53] have been frequently used for the design of fuzzy rule-based systems from numerical data under the name of genetic fuzzy systems [36, 37, 63] since the early 1990s [46, 101, 102, 123, 137, 139].

Fuzzy rule-based systems are universal approximators of nonlinear functions [106, 111, 143] in a similar way to neural networks [50, 67, 68]. Theoretically we can improve their approximation accuracy on training data to an arbitrarily specified level by increasing their complexity. Neuro-fuzzy systems and genetic fuzzy systems have been successfully used for such an accuracy improvement task. Accuracy improvement can be viewed as the following optimization problem:

$$\text{Maximize } \ Accuracy(S), \tag{1}$$

where S is a fuzzy rule-based system, and $Accuracy(S)$ is an accuracy measure (e.g., classification rate). This formulation can be also rewritten as

$$\text{Minimize } \ Error(S), \tag{2}$$

where $Error(S)$ is an error measure (e.g., misclassification rate).

Accuracy improvement of fuzzy rule-based systems usually leads to the increase in their complexity (i.e., the deterioration in their interpretability) because there is a tradeoff relation between accuracy maximization and complexity minimization. Such an accuracy-complexity tradeoff relation is illustrated in Figure 1 where the horizontal and vertical axes show a complexity measure and an error measure, respectively. Both measures are to be minimized in the design of fuzzy rule-based systems.

When we try to minimize the error measure in Figure 1, the complexity measure will be increased. This means that we may obtain an accurate fuzzy rule-based system with high complexity (i.e., with poor interpretability) through error minimization. On the other hand, the error measure will be increased when we try to minimize the complexity measure in Figure 1. This means that we may obtain an interpretable but inaccurate fuzzy rule-based system through complexity minimization. These discussions suggest that we cannot obtain an ideal fuzzy rule-based system (i.e., a very simple and very accurate system located around the origin of Figure 1).

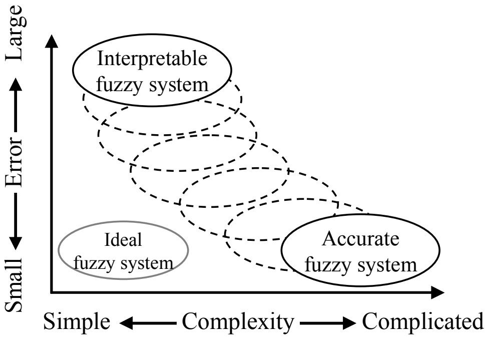


Fig. 1 Illustration of the tradeoff relation between accuracy maximization (i.e., error minimization) and complexity minimization in the design of fuzzy rule-based systems. Ellipsoids show non-dominated fuzzy rule-based systems along the accuracy-complexity tradeoff curve

As we have already explained, an undesirable side-effect of accuracy maximization of fuzzy rule-based systems is interpretability deterioration. Another problem of accuracy maximization is the overfitting of fuzzy rule-based systems to training data. In genetic fuzzy systems (and also in neuro-fuzzy systems), an accuracy measure in (1) or an error measure in (2) is usually defined for training data. As is well known in the field of statistical learning theory [30], the minimization of errors on training data does not always mean the minimization of errors on unseen test data. In Figure 2, we illustrate the overfitting to training data. By increasing the complexity of fuzzy rule-based systems, errors on training data are usually improved monotonically as shown by the solid curve in Figure 2. On the other hand, errors on unseen test data start to deteriorate by increasing the complexity beyond the optimal complexity S^* as shown by the dotted curve in Figure 2. Thus accuracy maximization for training data often deteriorates the generalization ability of fuzzy rule-based systems for unseen test data.

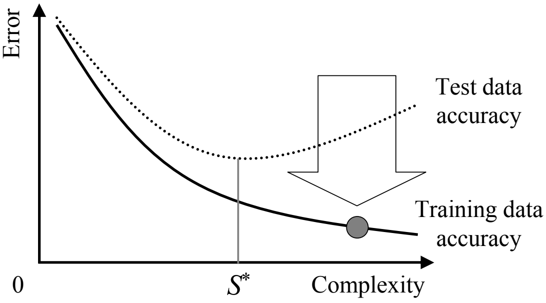


Fig. 2 Illustration of the overfitting to training data. The arrow shows the minimization of errors on training data

Since the late 1990s, the importance of interpretability maintenance in the design of fuzzy rule-based systems has been pointed out by many studies [4, 9, 22, 25, 69, 95, 100, 116, 130, 132, 133, 145]. In other words, complexity minimization as well as accuracy maximization was taken into account in order to design accurate and interpretable fuzzy rule-based systems. Whereas accuracy maximization and complexity minimization were simultaneously considered, the design of fuzzy rule-based systems was handled in the framework of single-objective optimization in those studies. That is, the two goals were integrated as follows:

$$\text{Optimize} \;\; f(S) = f(Accuracy(S), Complexity(S)), \tag{3}$$

where $f(S)$ is an integrated objective function (i.e., an integrated fitness function in genetic fuzzy systems), which combines an accuracy measure $Accuracy(S)$ and a complexity measure $Complexity(S)$. In some studies, the integrated objective function in (3) can be more appropriately rewritten as

$$\text{Optimize} \;\; f(S) = f(Accuracy(S), Interpretability(S)), \tag{4}$$

where $Interpretability(S)$ is an interpretability measure.

The following weighted sum objective function for fuzzy classifier design [86] is a typical example of the integrated objective function $f(S)$ in (3):

$$\text{Maximize} \;\; f(S) = w_1 \cdot Accuracy(S) - w_2 \cdot Complexity(S), \tag{5}$$

where $\mathbf{w} = (w_1, w_2)$ is a non-negative weight vector. The number of correctly classified training patterns and the number of fuzzy rules were used as an accuracy measure and a complexity measure, respectively. The weighted sum approach in [86] is one of the earliest studies where accuracy maximization and complexity minimization were explicitly performed by genetic algorithms in the design of fuzzy rule-based systems.

One difficulty in the weighted sum-based approach is that the specification of an appropriate weight vector is not easy and problem-dependent whereas the finally obtained fuzzy rule-based system strongly depends on its specification. For example, when the accuracy weight w_1 is too large in (5), we may obtain a too complicated fuzzy rule-based system as shown by A in Figure 3. On the other hand, when the complexity weight w_2 is too large, we may obtain a too simple fuzzy rule-based system as shown by B in Figure 3. Only when the weight vector is appropriately specified, we will obtain a fuzzy rule-based system with appropriate complexity (i.e., with high generalization ability for unseen test data). Almost all the above-mentioned studies with integrated objective functions share similar difficulties (i.e., it is not easy to specify an appropriate objective function for integrating accuracy maximization and complexity minimization).

As we have already explained, a single fuzzy rule-based system is obtained from an integrated objective function in (3) or (4) in single-objective approaches. On the

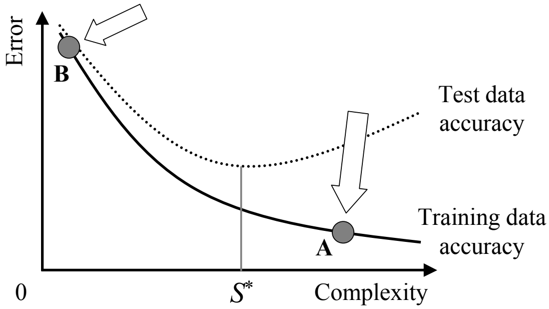


Fig. 3 Illustration of the weighted sum approach in (5). The circle A shows a complicated fuzzy rule-based system obtained when the accuracy weight is too large. The circle B shows a simple fuzzy rule-based system obtained when the complexity weight is too large

other hand, a large number of non-dominated fuzzy rule-based systems are obtained in multiobjective approaches by solving the following multiobjective problem:

$$\text{Maximize } \mathit{Accuracy}(S) \text{ and minimize } \mathit{Complexity}(S). \tag{6}$$

For example, a two-objective fuzzy rule selection method was proposed in [76] to search for non-dominated fuzzy classifiers with respect to the maximization of the number of correctly classified training patterns and the minimization of the number of selected fuzzy rules.

In [79], not only the number of fuzzy rules but also the total number of antecedent conditions (i.e., the total rule length) was minimized. In this case, the multiobjective problem in (6) can be rewritten as follows:

$$\text{Maximize } \mathit{Accuracy}(S), \text{ and minimize } \mathit{Complexity}_1(S) \text{ and } \mathit{Complexity}_2(S), \tag{7}$$

where $\mathit{Complexity}_1(S)$ and $\mathit{Complexity}_2(S)$ are different complexity measures.

The basic idea of multiobjective approaches is to search for a large number of non-dominated fuzzy rule-based systems with different tradeoffs between accuracy maximization and complexity minimization. This idea is illustrated in Figure 4 where multiple arrows show search directions for finding various non-dominated fuzzy rule-based systems with respect to error minimization and complexity minimization. Simple and inaccurate fuzzy rule-based systems are located in the upper left part of this figure while complicated and accurate ones are in the lower right part. There exist a large number of non-dominated fuzzy rule-based systems along the accuracy-complexity tradeoff curve. Multiobjective approaches try to search for those non-dominated fuzzy rule-based systems as many as possible as shown in Figure 5. Evolutionary multiobjective optimization (EMO) algorithms [32, 34, 41] are used for this task. In multiobjective approaches, it is assumed that a single non-dominated fuzzy rule-based system is chosen from a large number of obtained ones by human users based on their preference with respect to accuracy and complexity. Some users may choose a fuzzy rule-based system with the highest test data accuracy (i.e., with the optimal complexity S^* in terms of the generalization ability in Figure 5). Other users may choose simpler fuzzy rule-based systems with higher interpretability than S^*.

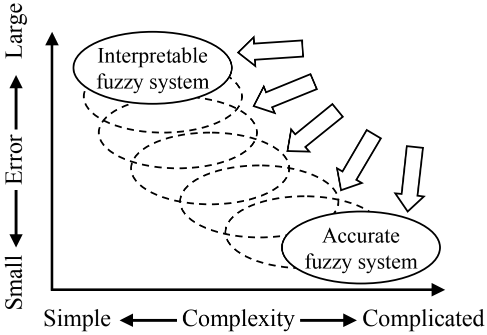


Fig. 4 Illustration of search directions for finding various non-dominated fuzzy rule-based systems with respect to accuracy maximization (i.e., error minimization) and complexity minimization in multiobjective approaches

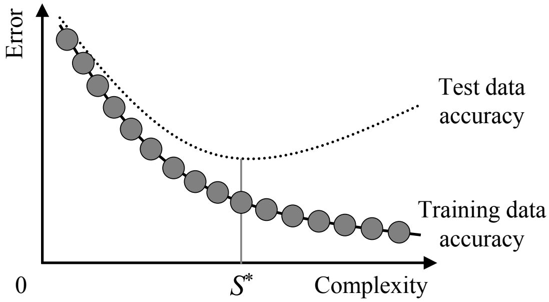


Fig. 5 A large number of non-dominated fuzzy rule-based systems along the accuracy-complexity tradeoff curve

Evolutionary multiobjective optimization (EMO), which has been used in multiobjective approaches to the design of fuzzy rule-based systems, is one of the most active research areas in the field of evolutionary computation. A large number of EMO algorithms have already been proposed in the literature [32, 34, 41]. Among them, NSGA-II [42], SPEA [147], and SPEA2 [146] are the most well-known and frequently-used EMO algorithms. Those EMO algorithms share some common ideas (with different implementation schemes) such as Pareto ranking-based fitness evaluation, diversity maintenance, and elitism.

The main advantage of EMO algorithms over non-evolutionary techniques in the field of multi-criteria decision making (MCDM [112]) is that a large number of non-dominated solutions with a wide range of objective values can be simultaneously obtained by their single run. On the contrary, multiple runs are required when we try to find a number of non-dominated solutions using MCDM techniques.

In this chapter, we explain a wide range of studies related to multiobjective genetic fuzzy systems in various research areas. Our explanation is, however, far from exhaustive because we try to cover broad research fields. See [36, 37, 63] for single-objective genetic fuzzy systems, [32, 34, 41] for evolutionary multiobjective optimization, [23, 24, 80] for interpretability-accuracy tradeoff of fuzzy rule-based systems, [97] for multiobjective machine learning, and [51] for multiobjective data mining.

2 Evolutionary Multiobjective Optimization

Before discussing multiobjective genetic fuzzy systems in the next section, we briefly explain evolutionary multiobjective optimization (EMO) algorithms together with some basic concepts in multiobjective optimization in this section.

2.1 Some Basic Concepts in Multiobjective Optimization

Let us consider the following *k*-objective maximization problem:

$$\text{Maximize } \mathbf{f}(\mathbf{y}) = (f_1(\mathbf{y}),\ f_2(\mathbf{y}),\ \ldots,\ f_k(\mathbf{y})), \tag{8}$$

$$\text{subject to} \;\; \mathbf{y} \in \mathbf{Y}, \tag{9}$$

where $\mathbf{f}(\mathbf{y})$ is the objective vector, $f_i(\mathbf{y})$ is the ith objective to be maximized, $\mathbf{y}$ is the decision vector, and $\mathbf{Y}$ is the feasible region in the decision space. Whereas the decision vector is usually denoted by $\mathbf{x}$ in various fields related to optimization, we use $\mathbf{y}$ in this section. This is because $\mathbf{x}$ is used to denote a pattern vector in the next section.

Let $\mathbf{y}$ and $\mathbf{z}$ be two feasible solutions of the k-objective maximization problem in (8) - (9). If the following conditions hold, $\mathbf{z}$ can be viewed as being better than $\mathbf{y}$:

$$^{\forall}i,\; f_i(\mathbf{y}) \leq f_i(\mathbf{z}) \;\; \text{and} \;\; ^{\exists}j,\; f_j(\mathbf{y}) < f_j(\mathbf{z}). \tag{10}$$

In this case, we say that $\mathbf{z}$ dominates $\mathbf{y}$ (equivalently $\mathbf{y}$ is dominated by $\mathbf{z}$).

When $\mathbf{y}$ is not dominated by any other feasible solutions (i.e., when there exists no feasible solution $\mathbf{z}$ that dominates $\mathbf{y}$), the solution $\mathbf{y}$ is referred to as a Pareto-optimal solution of the k-objective maximization problem in (8) - (9). The set of all Pareto-optimal solutions is called the Pareto-optimal solution set. The projection of the Pareto-optimal solution set forms the tradeoff surface in the objective space. This tradeoff surface is referred to as the Pareto front. Various EMO algorithms have already been proposed to efficiently find a set of non-dominated solutions that approximates the entire Pareto front [32, 34, 41, 42, 146, 147].

2.2 Evolutionary Multiobjective Optimization

Recently proposed evolutionary multiobjective optimization (EMO) algorithms such as NSGA-II [42] and SPEA [147] share the following three features:
(a) Use of Pareto dominance relation in (10) for fitness evaluation: Higher fitness values are assigned to non-dominated solutions in the current population than dominated ones.
(b) Use of the concept of crowding for fitness evaluation: Higher fitness values are assigned to solutions in uncrowded regions of the objective space than those in crowded regions.
(c) Use of non-dominated solutions as elite solutions: Non-dominated solutions in the current population are handled as elite solutions. That is, they are inherited to the next population with no modifications (or they are stored in an archive population separately from the next population).

As a representative algorithm, we explain the most well-known and frequently-used EMO algorithm in the literature: NSGA-II (elitist non-dominated sorting genetic algorithm) of Deb et al. [42]. Let P and N_{pop} be the current population and the population size, respectively (i.e., $N_{\text{pop}} = |P|$). Then the outline of NSGA-II can be written as follows:

```
Step 1:  $P :=$ Initialize($P$)
Step 2:  while a termination condition is not satisfied, do
Step 3:          $P' :=$ Selection($P$)
Step 4:          $P'' :=$ Genetic Operations($P'$)
Step 5:         $P := \text{Replace}(P \cup P'')$
Step 6:  end while
Step 7:  return (non-dominated solutions($P$))
```

First, N_{pop} solutions are randomly generated to form an initial population P in Step 1 as in standard single-objective genetic algorithms (SOGAs). Next, N_{pop} pairs of parents are selected from the current population P to form a parent population P' in Step 3. Binary tournament selection with replacement is used for parent selection. Then an offspring population P'' is constructed in Step 4 by generating a single offspring solution from each pair of parents in P' by crossover and mutation. Genetic operations in Step 4 are the same as those in standard SOGAs. The next population is constructed in Step 5 by choosing the best N_{pop} solutions from the $2 \cdot N_{\text{pop}}$ solutions in the current population P and the offspring population P''. Parent selection in Step 3 and generation update in Step 5 of NSGA-II are different from standard SOGAs. Steps 3-5 are iterated until a prespecified termination condition is satisfied. When the execution of NSGA-II is terminated, non-dominated solutions in the final population are presented in Step 7.

Each solution in the current population in Step 3 is evaluated by non-dominated sorting and a crowding distance. Solutions in the current population are sorted by Pareto dominance relation in (10) in the following manner (see Figure 6). First, Rank 1 is assigned to non-dominated solutions in the current population. Rank 1 solutions are tentatively removed from the current population. Next, Rank 2 is assigned to non-dominated solutions in the remaining population. Rank 2 solutions are tentatively removed from the remaining population. In this manner, solutions in the current population are sorted to assign a rank to each solution. The rank of each solution is used as the primary criterion in fitness evaluation.

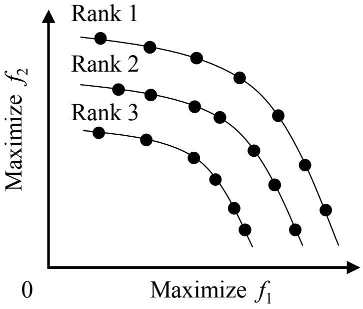


Fig. 6 Illustration of non-dominated sorting in NSGA-II

Solutions with the same rank are compared with each other using a secondary criterion called a crowding distance. In the calculation of the crowding distance for a solution, all solutions with the same rank are projected to each objective. Then the distance between its two adjacent solutions in the projected single-dimensional space is calculated for each objective. The crowding distance is the sum of the calculated distances over all objectives. When a solution has the maximum or minimum value of at least one objective among solutions with the same rank, an infinitely large

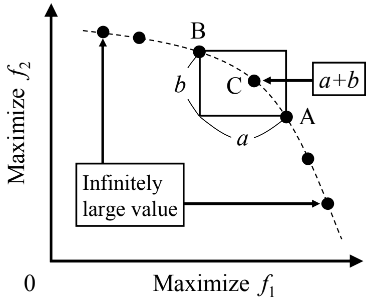


Fig. 7 Illustration of the calculation of the crowding distance

value is assigned to that solution as its crowding distance because it has only a single adjacent solution for at least one projected single-dimensional objective space. Figure 7 illustrates the calculation of the crowding distance for the case of two objectives. In Figure 7, an infinitely large value is assigned to two extreme solutions. The crowding distance of solution C is calculated as $a+b$ which is the Manhattan distance between its two adjacent solutions A and B.

In Step 5, each solution in the merged population (i.e., the current and offspring populations) is evaluated for generation update in the same manner as parent selection in Step 3 (i.e., by the non-dominated sorting and the crowded distance in the merged population). The best N_{pop} solutions are chosen from the merged population with $2 \cdot N_{\text{pop}}$ solutions to form the next population.

In NSGA-II, Pareto dominance-based fitness assignment is realized through the non-dominated sorting in Figure 6. Better ranks are assigned to better solutions with respect to Pareto dominance relation. This fitness assignment scheme generates the selection pressure toward the Pareto front. On the other hand, the crowding distance in Figure 7 is used as the secondary criterion to differentiate solutions with the same rank. The secondary criterion works as a diversity maintenance mechanism in NSGA-II in order to evenly distribute solutions over the entire Pareto front. Roughly speaking, the crowding distance-based secondary criterion widens the population along the Pareto front while the non-dominated sorting-based primary criterion pushes it toward the Pareto front.

On the other hand, elitism is implemented in the framework of the $(\mu+\lambda)$-ES generation update scheme in NSGA-II where $\mu=\lambda$. That is, the best μ solutions are chosen from $(\mu+\lambda)$ solutions in the current and offspring populations in order to form the next population. It is widely recognized that elitist EMO algorithms outperform non-elitist EMO algorithms [147]. Elitism usually has positive effects on both the convergence of solutions toward the Pareto front and the diversity along the Pareto front in EMO algorithms.

In Figure 8, we illustrate multiobjective evolution by NSGA-II on a two-objective 100-item knapsack problem. Figure 8 shows an initial population, an intermediate population at the 20th generation, and the final population at the 2000th generation. We can see from Figure 8 that the population moves toward the Pareto front while increasing the diversity of solutions. It should be noted that a large number of non-dominated solutions can be obtained by a single run of NSGA-II. This is the main advantage of EMO algorithms over other techniques for multiobjective optimization.

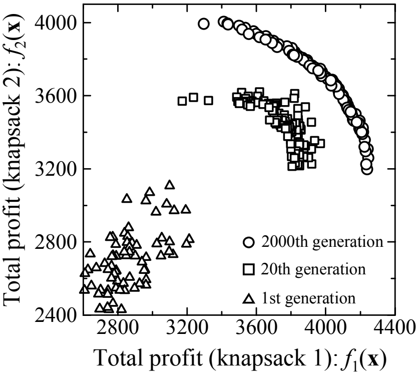


Fig. 8 Experimental results by a single run of NSGA-II on a two-objective 100-item knapsack problem. Parameters in NSGA-II were specified as follows: The population size was 100, the crossover probability was 0.8 (uniform crossover), the mutation probability was 0.05, and the termination condition was 2000 generations

3 Multiobjective Genetic Fuzzy Systems

In this section, we give a brief overview on multiobjective design of fuzzy rule-based systems. The use of multiobjective genetic algorithms for the design of fuzzy rule-based systems was first proposed in the mid-1990s by Ishibuchi et al. [75, 76]. They applied a two-objective genetic algorithm to a fuzzy rule selection problem where each fuzzy rule-based classifier S was evaluated by the following two objectives:

$f_1(S)$: The number of correctly classified training patterns by S,
$f_2(S)$: The number of fuzzy rules in S.

The first objective is an accuracy measure to be maximized while the second objective is a complexity measure to be minimized. Of course, we can use the average classification rate as an accuracy measure instead of the number of correctly classified training patterns. We can also use the average misclassification rate as an error measure to be minimized. Each solution (i.e., each fuzzy rule-based classifier S) of the two-objective fuzzy rule selection problem is a subset of a large number of candidate rules, which is represented by a binary string. The string length is the same as the total number of candidate rules. Each bit value shows the inclusion or exclusion of the corresponding candidate rule.

The two-objective formulation was extended to a three-objective problem [79] where the following complexity measure was included:

$f_3(S)$: The total number of antecedent conditions of fuzzy rules in S.

Fuzzy rules with many *don't care* conditions were used in [79] to handle high-dimensional classification problems. It should be noted that *don't care* conditions are not counted as antecedent conditions in $f_3(S)$. For example, the antecedent part "If x_1 is *don't care* and x_2 is *small* and x_3 is *don't care* and x_4 is *large*" is viewed as "If x_2 is *small* and x_4 is *large*" with two antecedent conditions. In [89], a multiobjective genetic local search algorithm was applied to the three-objective fuzzy rule selection problem with $f_1(S)$, $f_2(S)$ and $f_3(S)$. In [81], a three-objective genetic algorithm was used for non-fuzzy rule selection. The same three objectives were also used in multiobjective fuzzy genetics-based machine learning (GBML) algorithms in [79, 85].

Recently multiobjective approaches have been used for the design of fuzzy rule-based systems in many studies. As SOGAs in single-objective genetic fuzzy systems [36, 37, 63], EMO algorithms can optimize various aspects of fuzzy rule-based systems (e.g., input selection, rule generation, rule selection, determination of the number of fuzzy sets for each variable, optimization of the shape of each fuzzy set, etc.) in multiobjective genetic fuzzy systems. Various criteria for evaluating fuzzy rule-based systems can be also used as objectives in multiobjective genetic fuzzy systems.

For example, the number of fuzzy rules and the total number of antecedent conditions were used together with an accuracy measure for multiobjective design of fuzzy rule-based classifiers in [124, 134, 138] as in [79, 81, 85]. That is, EMO algorithms were applied to three-objective problems where $f_2(S)$ and $f_3(S)$ were used as complexity measures. These two complexity measures were often used to avoid the overfitting to training data in multiobjective design of fuzzy rule-based classifiers.

Multiobjective approaches have been used not only for classification problems but also for function approximation problems. The above-mentioned two complexity measures were used for classification problems (together with the average misclassification rate) and function approximation problems (together with the mean squared error) in Wang et al. [140]. The total number of fuzzy sets instead of the total number of antecedent conditions was used together with an accuracy measure and the number of fuzzy rules in a three-objective formulation for function approximation problems in Xing et al. [144]. On the other hand, Alcala et al. [10] and Gonzalez et al. [57] used a two-objective formulation with the mean squared error and the number of fuzzy rules for function approximation problems. Jimenez et al. [94] and Gomez-Skarmeta [54] used a different two-objective formulation for function approximation problems. In their two-objective formulation, one objective is the mean squared error and the other objective is an interpretability measure defined by the similarity between adjacent fuzzy sets.

In some studies, more than three objectives were used for multiobjective design of fuzzy rule-based systems. For example, the following five objectives were used in Wang et al. [141]: accuracy, completeness and distinguishability, non-redundancy, the number of fuzzy rules, and the total number of fuzzy sets. It should be noted that "completeness and distinguishability" was used as a single objective (for details, see Wang et al. [141]).

In all the above-mentioned studies [10, 54, 57, 75, 76, 79, 81, 85, 89, 94, 124, 134, 138, 140, 141, 144] on multiobjective genetic fuzzy systems for classification and function approximation problems, multiobjective problems were formulated using both accuracy and complexity measures. That is, EMO algorithms in these studies were used to search for a number of non-dominated fuzzy rule-based systems with different accuracy-complexity tradeoffs. On the other hand, in multiobjective genetic fuzzy systems for control problems, multiple performance measures together with no complexity measures were often used. For example, Stewart et al. [135] handled the design of fuzzy logic controllers as a three-objective problem where three objectives were the current tracking error, the velocity tracking error, and the power consumption. Chen and Chiang [29] formulated a three-objective problem using the number of collisions, the distance between the target and lead points of the new path, and the number of explored actions. Kim and Roschke [105] used the following two performance measures: the structural acceleration and the base displacement level.

Whereas both complexity and accuracy have been taken into account in many studies on multiobjective genetic fuzzy systems for classification and function approximation problems, only accuracy measures were used for fuzzy control in [29, 105, 135]. This is partially because multiple performance measures are usually involved in controllers while the performance of classifiers and function approximators can be often evaluated by a single accuracy measure. Another possible reason is that the overfitting to training data is not so critical in control problems if compared with classification and function approximation problems. Moreover, the number of input variables is usually much larger in classification and function approximation problems than in control problems. This is also a possible reason why complexity minimization (including input selection) has been more widely used in classification and function approximation problems than in control problems. Of course, it is possible to use multiple accuracy measures for classification problems with no complexity measures (e.g., the false negative rate and the false positive rate). It is also possible to use complexity measures for control problems together with accuracy measures. In general, multiple accuracy measures and multiple complexity measures are involved in classification, function approximation and control problems. The choice of an appropriate set of objective functions is an important future research issue.

The overview on multiobjective genetic fuzzy systems in this subsection is far from complete. Our overview is limited to journal papers. For a more complete list of references including conference papers, see Evolutionary Multiobjective Optimization of Fuzzy Rule-Based Systems Bibliography Page (http://www2.ing.unipi.it/˜o613499/emofrbss.html).

4 Multiobjective Genetic Fuzzy Rule Selection

As we have already explained, the first study on multiobjective genetic fuzzy systems was multiobjective genetic fuzzy rule selection [75, 76]. In this section, we explain how EMO algorithms can be used for multiobjective genetic fuzzy rule selection. Through computational experiments, we demonstrate that a large number of non-dominated fuzzy rule-based classifiers along the accuracy-complexity tradeoff surface can be found by a single run of an EMO algorithm.

4.1 Fuzzy Rule-Based Classifiers

Let us assume that we have m training (i.e., labeled) patterns $\mathbf{x}_p = (x_{p1}, \ldots, x_{pn})$, $p = 1, 2, \ldots, m$ from M classes in an n-dimensional pattern space where x_{pi} is the attribute value of the pth pattern for the ith attribute ($i = 1, 2, \ldots, n$). For the simplicity of explanation, we assume that all the attribute values have already been normalized into real numbers in the unit interval [0, 1]. Thus the pattern space of our classification problem is an n-dimensional unit-hypercube $[0, 1]^n$.

For our n-dimensional pattern classification problem, we use fuzzy rules of the following type:

$$\text{Rule } R_q: \text{ If } x_1 \text{ is } A_{q1} \text{ and ... and } x_n \text{ is } A_{qn} \text{ then Class } C_q \text{ with } CF_q, \tag{11}$$

where R_q is the label of the qth fuzzy rule, $\mathbf{x} = (x_1, ..., x_n)$ is an n-dimensional pattern vector, A_{qi} is an antecedent fuzzy set ($i = 1, 2, ..., n$), C_q is a class label, and CF_q is a rule weight. We denote the antecedent fuzzy sets of R_q as a fuzzy vector $\mathbf{A}_q = (A_{q1}, A_{q2}, ..., A_{qn})$.

We use 14 fuzzy sets in four fuzzy partitions with different granularities in Figure 9. In addition to those 14 fuzzy sets, we also use the domain interval [0, 1] itself as an antecedent fuzzy set in order to represent a *don't care* condition. Whereas we use the prespecified membership functions with no further adjustment in this chapter, we can include a learning mechanism of the membership functions in multiobjective genetic fuzzy rule selection. The number of antecedent fuzzy sets for each attribute (i.e., granularity of each attribute) can be also handled as a decision variable in multiobjective genetic fuzzy rule selection.

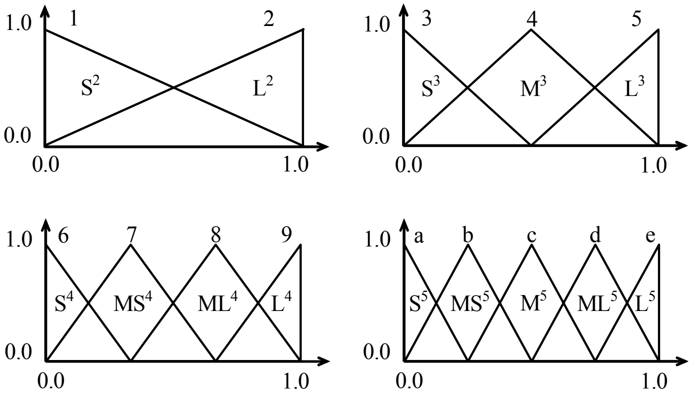


Fig. 9 Fourteen antecedent fuzzy sets used in this chapter

Let S be a set of fuzzy rules of the form in (11). When an input pattern $\mathbf{x}_p$ is to be classified by S, first we calculate the compatibility grade of $\mathbf{x}_p$ with the antecedent part $\mathbf{A}_q = (A_{q1}, A_{q2}, ..., A_{qn})$ of each fuzzy rule R_q in S using the product operation as

$$\mu_{\mathbf{A}_q}(\mathbf{x}_p) = \mu_{A_{q1}}(x_{p1}) \cdot ... \cdot \mu_{A_{qn}}(x_{pn}), \tag{12}$$

where $\mu_{A_{qi}}(\cdot)$ is the membership function of the antecedent fuzzy set A_{qi}. Then a single winner rule R_w is identified using the compatibility grade and the rule weight of each fuzzy rule as

$$\mu_{\mathbf{A}_w}(\mathbf{x}_p) \cdot CF_w = \max\{\mu_{\mathbf{A}_q}(\mathbf{x}_p) \cdot CF_q \mid R_q \in S\}. \tag{13}$$

The input pattern $\mathbf{x}_p$ is classified as the consequent class C_w of the winner rule R_w. When multiple fuzzy rules with different consequent classes have the same maximum value in (13), the classification of $\mathbf{x}_p$ is rejected. If there is no compatible fuzzy rule with $\mathbf{x}_p$, its classification is also rejected.

4.2 Candidate Rule Generation

Multiobjective genetic fuzzy rule selection is a two-step method. In the first step, a prespecified number of promising fuzzy rules are generated from training patterns as candidate rules. In the second step, an EMO algorithm is used to search for non-dominated fuzzy rule-based classifiers (i.e., non-dominated subsets of the generated candidate rules). In this subsection, we explain the first step.

Since we use the 14 antecedent fuzzy sets in Figure 9 and an additional *don't care* fuzzy set [0, 1] for each attribute of our n-dimensional classification problem, the total number of possible fuzzy rules is 15^n. Among these possible rules, we examine only short fuzzy rules with a small number of antecedent conditions (i.e., short fuzzy rules with many *don't care* conditions) to generate candidate rules. In this chapter, we examine only short fuzzy rules with three or less antecedent conditions.

The consequent class C_q and the rule weight CF_q of each fuzzy rule R_q are specified from training patterns compatible with its antecedent part $\mathbf{A}_q = (A_{q1}, A_{q2}, ..., A_{qn})$ in the following heuristic manner [80]. First we calculate the confidence of each class for the antecedent part $\mathbf{A}_q$ as

$$c(\mathbf{A}_q \Rightarrow \text{Class } h) = \frac{\sum_{\mathbf{x}_p \in \text{Class } h} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}{\sum_{p=1}^{m} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}, \quad h = 1, 2, ..., M. \tag{14}$$

It should be noted that "$\mathbf{A}_q \Rightarrow \text{Class } h$" means the fuzzy rule with the antecedent part $\mathbf{A}_q$ and the consequent class h. Then the consequent class C_q is specified by identifying the class with the maximum confidence:

$$c(\mathbf{A}_q \Rightarrow \text{Class } C_q) = \max_{h=1,2,...,M} \{c(\mathbf{A}_q \Rightarrow \text{Class } h)\}. \tag{15}$$

In this manner, we generate the fuzzy rule R_q (i.e., $\mathbf{A}_q \Rightarrow \text{Class } C_q$) with the antecedent part $\mathbf{A}_q$ and the consequent class C_q. We do not generate any fuzzy rules with the antecedent part $\mathbf{A}_q$ if there is no compatible training pattern with $\mathbf{A}_q$.

The rule weight CF_q of each fuzzy rule R_q has a large effect on the performance of fuzzy rule-based classifiers. We use the following specification of CF_q because good results were reported in the literature [90]:

$$CF_q = c(\mathbf{A}_q \Rightarrow \text{Class}\, C_q) - \sum_{\substack{h=1 \\ h \neq C_q}}^{M} c(\mathbf{A}_q \Rightarrow \text{Class}\, h). \tag{16}$$

We do not use the fuzzy rule R_q as a candidate rule if the rule weight CF_q is not positive (i.e., if its confidence is not larger than 0.5). Whereas we heuristically specify the rule weight of each fuzzy rule by (16) in this chapter, we can include a learning mechanism of the rule weight in multiobjective genetic fuzzy rule selection.

In the above-mentioned heuristic manner, we can generate a large number of short fuzzy rules as candidate rules in multiobjective fuzzy rule selection. An EMO algorithm is used to search for non-dominated subsets of the generated candidate rules. When the number of candidate rules is too large (e.g., tens of thousands), it is not easy for EMO algorithms to efficiently perform multiobjective fuzzy rule selection. Thus we use a prescreening procedure to decrease the number of candidate rules. Our prescreening procedure is based on well-known rule evaluation measures in the field of data mining [7]: *support* and *confidence*.

The confidence of a rule evaluates the accuracy of the association from the antecedent part to the consequent part. We have already shown a fuzzy version of the confidence in (14). On the other hand, the support indicates the percentage of covered patterns. Its fuzzy version can be written as follows [80]:

$$s(R_q) = s(\mathbf{A}_q \Rightarrow \text{Class } C_q) = \frac{\sum\limits_{\mathbf{x}_p \in \text{ Class } C_q} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}{m}. \tag{17}$$

For prescreening candidate rules, we use two threshold values: the minimum support and the minimum confidence. We exclude fuzzy rules that do not satisfy these two threshold values. Among short fuzzy rules satisfying these two threshold values, we choose a prespecified number of candidate rules for each class. As a rule evaluation criterion, we use the product of the support $s(R_q)$ and the confidence $c(R_q)$. That is, we choose a prespecified number of the best candidate rules for each class with respect to $s(R_q) \cdot c(R_q)$.

4.3 Multiobjective Fuzzy Rule Selection

Let us assume that we have N candidate rules (i.e., N/M candidate rules for each of M classes). Any subset S of the N candidate rules can be represented by a binary string of length N: $S = s_1 s_2 ... s_N$ where $s_j = 1$ and $s_j = 0$ mean the inclusion and the exclusion of the jth candidate rule R_j in the subset S, respectively (j = 1, 2, ..., N). Such a binary string S is used as an individual in an EMO algorithm for multiobjective fuzzy rule selection. It should be noted that S can be viewed as a fuzzy rule-based classifier.

Each fuzzy rule-based classifier S is evaluated by the three objectives explained in the previous section (i.e., $f_1(S)$: the number of correctly classified training patterns, $f_2(S)$: the number of selected fuzzy rules, and $f_3(S)$: the total number of antecedent conditions). That is, our multiobjective fuzzy rule selection problem is written as

$$\text{Maximize } f_1(S), \text{ and minimize } f_2(S) \text{ and } f_3(S). \tag{18}$$

We use NSGA-II of Deb et al. [42] to search for non-dominated fuzzy rule-based classifiers with respect to these three objectives. Of course, we can use other EMO algorithms. Since each individual is represented by a binary string, we can use standard genetic operations for binary strings in NSGA-II for multiobjective fuzzy rule selection. In our computational experiments, uniform crossover and bit-flip mutation were used in NSGA-II. The execution of NSGA-II was terminated at the prespecified number of generations.

In order to efficiently decrease the number of fuzzy rules in each rule set S, we can use two heuristic techniques. One is biased mutation where a larger mutation probability is assigned to the mutation from 1 to 0 than that from 0 to 1. The other is the removal of unnecessary fuzzy rules. Since we use the single winner-based scheme in (13) for classifying each training pattern by a fuzzy rule-based classifier S, some fuzzy rules in S may classify no training patterns. We can remove those unnecessary fuzzy rules from S without changing any classification results by S (i.e., without changing the first objective $f_1(S)$). At the same time, the removal of those unnecessary fuzzy rules improves the second objective $f_2(S)$ and the third objective $f_3(S)$. In our computational experiments, the necessity of each fuzzy rule in S was checked when $f_1(S)$ was calculated. All the unnecessary fuzzy rules were removed from S before $f_2(S)$ and $f_3(S)$ were calculated. This heuristic procedure can be viewed as a kind of local search since $f_2(S)$ and $f_3(S)$ are improved without deteriorating $f_1(S)$.

4.4 Computational Experiments

In our computational experiments, we used five data sets in Table 1 from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/). Before performing multiobjective fuzzy rule selection, we normalized all attribute values in the data sets to real numbers in the unit interval [0, 1].

We divided each data set into two subsets of the same size: training data and test data. Using the training data, first we extracted fuzzy rules satisfying the minimum support and the minimum confidence in Table 1.

The upper bound on the number of antecedent conditions of each candidate fuzzy rule was specified as three for all data sets. Among qualified fuzzy rules satisfying

Table 1 Five data sets used in our computational experiments. Rule extraction criteria (i.e., specified values of the minimum support and confidence) are also shown for each data set

Data set	Attributes	Patterns	Classes	Support	Confidence
Breast W	9	683†	2	0.01	0.6
Glass	9	214	6	0.001	0.6
Heart C	13	297†	5	0.001	0.6
Iris	4	150	3	0.01	0.6
Wine	13	178	3	0.04	0.6

† Incomplete patterns with missing values are not included.

the minimum confidence, the minimum support and the maximum number of antecedent conditions, we chose the best 300 rules for each class using the product of support and confidence. Those 300 fuzzy rules for each class were used as candidate rules in multiobjective fuzzy rule selection.

Then we applied NSGA-II to the candidate rules to search for non-dominated fuzzy rule-based classifiers with respect to the three objectives in (18). We used the following parameter specifications in NSGA-II:

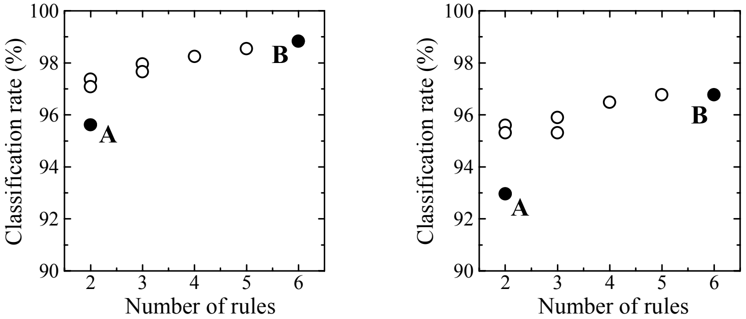
Population size: 200 strings,
Crossover probability: 0.9 (uniform crossover),
Mutation probability: 0.05 (1 → 0) and $1/N$ (0 → 1) where N is the string length,
Termination condition: 5000 generations.

In the following, we report experimental results of multiobjective fuzzy rule selection on each data set.

Wisconsin Breast Cancer Data Set: First we randomly divided the data set into 342 and 341 patterns for training and testing, respectively. Next we generated 76270 fuzzy rules of length three or less from the 342 training patterns, which satisfied the minimum support 0.01 and the minimum confidence 0.6. Among those fuzzy rules, we chose 300 candidate rules for each class using the product of support and confidence as the rule evaluation criterion. Then we applied NSGA-II to the candidate rules for multiobjective fuzzy rule selection. From its single run, 11 non-dominated fuzzy classifiers (i.e., 11 non-dominated subsets of the candidate rules) were obtained. Finally each of the obtained fuzzy rule-based classifier was evaluated for the training patterns and the test patterns.

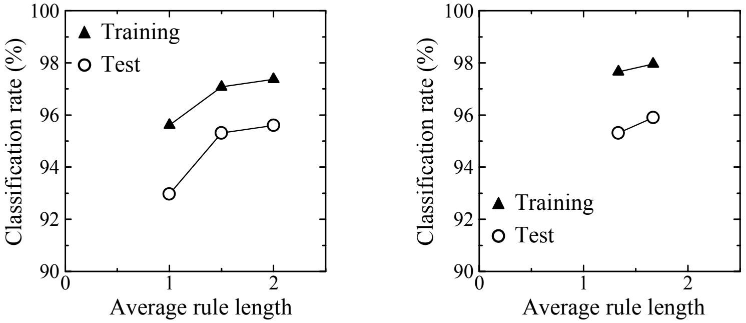
Training data accuracy and test data accuracy of each fuzzy rule-based classifier are shown in Figure 10 (a) and Figure 10 (b), respectively. Each circle in Figure 10 shows a fuzzy rule-based classifier. Some of the obtained fuzzy rule-based classifiers (e.g., a fuzzy rule-based classifier with only a single fuzzy rule) are not shown because their classification rates are out of the range of the vertical axis of each plot in Figure 10. Since we used not only the number of fuzzy rules but also the total number of antecedent conditions as complexity measures in multiobjective fuzzy rule selection, different fuzzy rule-based classifiers with the same number of fuzzy rules were obtained in Figure 10. For example, fuzzy rule-based classifiers with two and three fuzzy rules are shown in Figure 11. It should be noted that the horizontal axis is the average rule length (i.e., the average number of antecedent conditions in each fuzzy rule) in Figure 11 whereas it is the number of fuzzy rules in Figure 10.

We can observe a clear accuracy-complexity tradeoff relation in Figure 10 (a) for the training data. The classification rate on the training data increases with the increase in the number of fuzzy rules in Figure 10 (a). This means that we cannot simultaneously achieve the accuracy maximization and the complexity minimization. A similar tradeoff relation is observed in Figure 10 (b) for the test data. No clear deterioration in the generalization ability due to the increase in the complexity of fuzzy rule-based classifiers is observed in Figure 10 (b). That is, we observe no clear indication of the overfitting of fuzzy rule-based systems to the training data in



(a) Training data accuracy. (b) Test data accuracy.

Fig. 10 Obtained non-dominated fuzzy rule-based classifiers (Breast W)



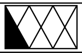
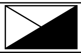
(a) Classifiers with two rules. (b) Classifiers with three rules.

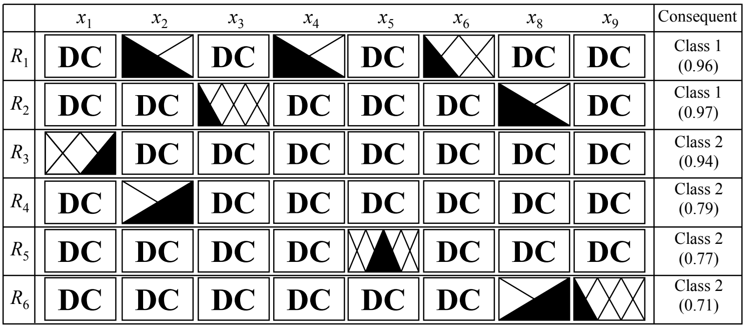
Fig. 11 Relation between the accuracy and the average rule length (Breast W)

Figure 10 (b). We can also observe a clear tradeoff relation between the accuracy of fuzzy rule-based classifiers and the average rule length in Figure 11 for both the training data and the test data. That is, there exists the accuracy-complexity tradeoff relation when the average rule length is used as a complexity measure in Figure 11.

Two fuzzy rule-based classifies A and B in Figure 10 are shown in Figure 12 and Figure 13, respectively. The classifier A in Figure 12 with two fuzzy rules is the simplest one with the highest interpretability in Figure 10 whereas the classifier B with six fuzzy rules is the most complicated one with the highest training data accuracy. There exist many fuzzy rule-based classifiers between these two extremes. The classifier B may be chosen by many human users since it also has the highest generalization ability in Figure 10 (b). Some human users, however, may choose

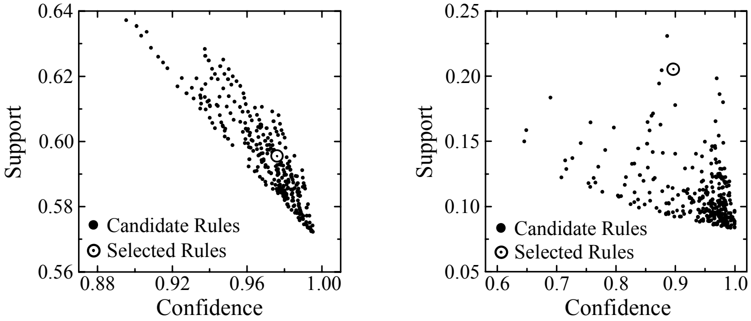
Fig. 12 The simplest fuzzy rule-based classifier A in Figure 10 (Breast W)

	x_2	x_3	Consequent
R_1	DC		Class 1 (0.95)
R_2		DC	Class 2 (0.79)



	x_1	x_2	x_3	x_4	x_5	x_6	x_8	x_9	Consequent
R_1	DC		DC		DC		DC	DC	Class 1 (0.96)
R_2	DC	DC		DC	DC	DC		DC	Class 1 (0.97)
R_3		DC	DC	DC	DC	DC	DC	DC	Class 2 (0.94)
R_4	DC		DC	DC	DC	DC	DC	DC	Class 2 (0.79)
R_5	DC	DC	DC	DC		DC	DC	DC	Class 2 (0.77)
R_6	DC	DC	DC	DC	DC	DC			Class 2 (0.71)

Fig. 13 The most complicated fuzzy rule-based classifier B in Figure 10 (Breast W)



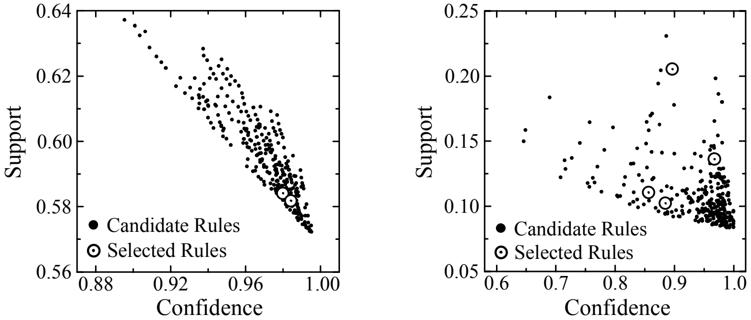
(a) Class 1. (b) Class 2.

Fig. 14 Fuzzy rules in Figure 12 included in the classifier A in Figure 10 (Breast W)

simpler ones than the classifier B in the situation where not only the accuracy but also the interpretability is an important criterion.

In Figure 14, the two fuzzy rules in Figure 12 are shown in the confidence-support space together with the other candidate rules. We can see from Figure 14 that these two fuzzy rules have large support values. This means that they cover many training patterns. Usually simple fuzzy rule-based classifiers consist of a small number of general fuzzy rules that cover many training patterns. On the other hand, the six fuzzy rules in Figure 13 are shown in Figure 15. The most complicated fuzzy rule-based classifier B includes not only general fuzzy rules with large support values but also specific fuzzy rules with small support values and high confidence values.

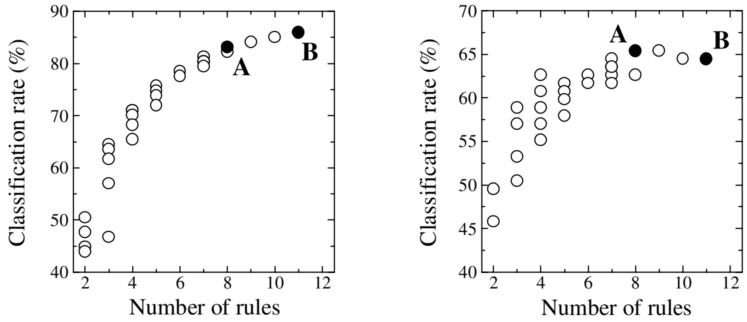
Glass Data Set: Experimental results are shown in Figure 16. Multiobjective fuzzy rule selection found 34 non-dominated fuzzy rule-based classifiers (some of them are not shown in Figure 16 due to their low accuracy). We can observe a clear accuracy-complexity tradeoff relation in Figure 16 (a) for training data. We can also see that the fuzzy rule-based classifier with the best training data accuracy does not always have the best test data accuracy (Figure 16 (b)). We examined the two fuzzy rule-based classifiers A and B in Figure 16 in detail. One interesting observation is



(a) Class 1.

(b) Class 2.

Fig. 15 Fuzzy rules in Figure 13 included in the classifier B in Figure 10 (Breast W)



(a) Training data accuracy.

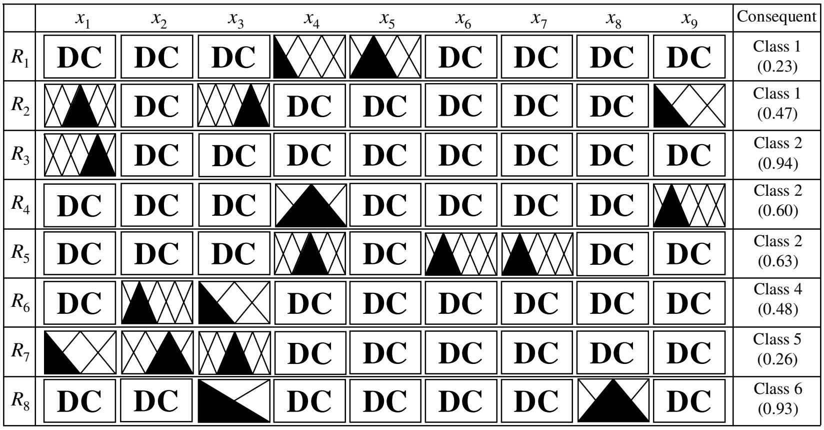
(b) Test data accuracy.

Fig. 16 Obtained non-dominated fuzzy rule-based classifiers (Glass)

that the classifier A with the highest test data accuracy in Figure 16 (b) has no fuzzy rule with Class 3 consequent as shown in Figure 17. On the other hand, the classifier B with the highest training data accuracy in Figure 16 (a) has at least one fuzzy rule for each class. The test data accuracy of the classifier B is, however, inferior to the classifier A due to the overfitting to the training data.

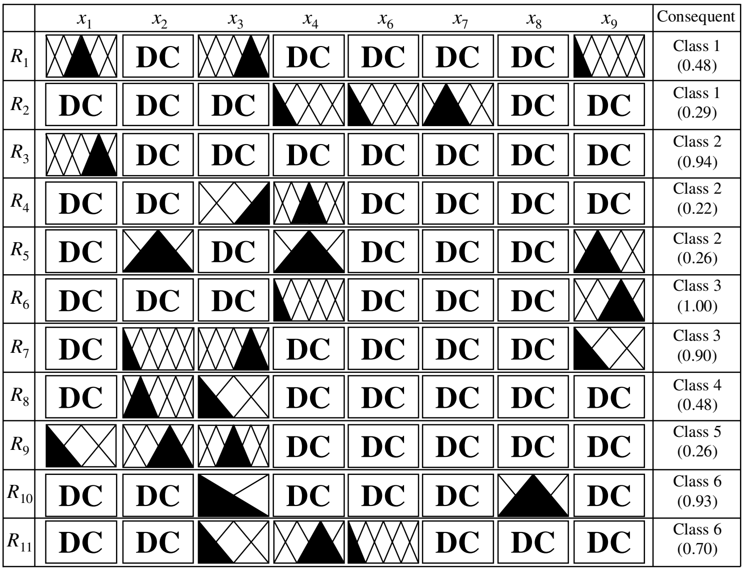
Cleveland Heart Disease Data Set: Experimental results are shown in Figure 19. Multiobjective fuzzy rule selection found 48 non-dominated fuzzy rule-based classifiers. We can observe a clear accuracy-complexity tradeoff relation in Figure 19 (a) for training data. On the other hand, we can observe a clear indication of the overfitting of fuzzy rule-based classifiers to training data due to the increase in their complexity in Figure 19 (b). That is, the test data accuracy decreases with the increase in the number of fuzzy rules in the range with more than five fuzzy rules.

In Figure 20, we show the relation between the accuracy and the average rule length of the obtained fuzzy rule-based classifiers with three rules (a) and four rules (b). We can observe a clear deterioration of the test data accuracy due to the increase in the average rule length in Figure 20. One may notice that Figure 20 (b) includes



	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	Consequent
R_1	DC	DC	DC			DC	DC	DC	DC	Class 1 (0.23)
R_2		DC		DC	DC	DC	DC	DC		Class 1 (0.47)
R_3		DC	DC	DC	DC	DC	DC	DC	DC	Class 2 (0.94)
R_4	DC	DC	DC		DC	DC	DC	DC		Class 2 (0.60)
R_5	DC	DC	DC		DC			DC	DC	Class 2 (0.63)
R_6	DC			DC	DC	DC	DC	DC	DC	Class 4 (0.48)
R_7				DC	DC	DC	DC	DC	DC	Class 5 (0.26)
R_8	DC	DC		DC	DC	DC	DC		DC	Class 6 (0.93)

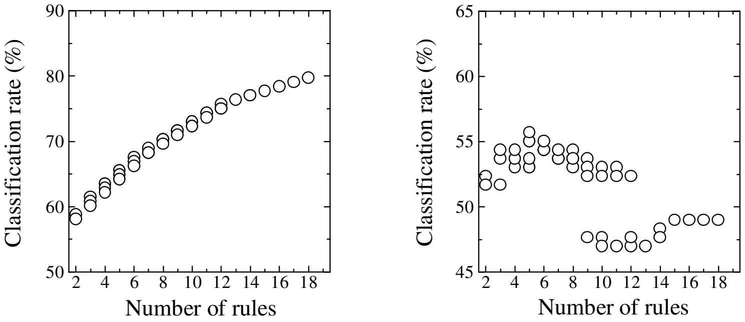
Fig. 17 Fuzzy rule-based classifier A with the highest test data accuracy in Figure 16 (Glass)



	x_1	x_2	x_3	x_4	x_6	x_7	x_8	x_9	Consequent
R_1		DC		DC	DC	DC	DC		Class 1 (0.48)
R_2	DC	DC	DC				DC	DC	Class 1 (0.29)
R_3		DC	DC	DC	DC	DC	DC	DC	Class 2 (0.94)
R_4	DC	DC			DC	DC	DC	DC	Class 2 (0.22)
R_5	DC		DC		DC	DC	DC		Class 2 (0.26)
R_6	DC	DC	DC		DC	DC	DC		Class 3 (1.00)
R_7	DC			DC	DC	DC	DC		Class 3 (0.90)
R_8	DC			DC	DC	DC	DC	DC	Class 4 (0.48)
R_9				DC	DC	DC	DC	DC	Class 5 (0.26)
R_{10}	DC	DC		DC	DC	DC		DC	Class 6 (0.93)
R_{11}	DC	DC				DC	DC	DC	Class 6 (0.70)

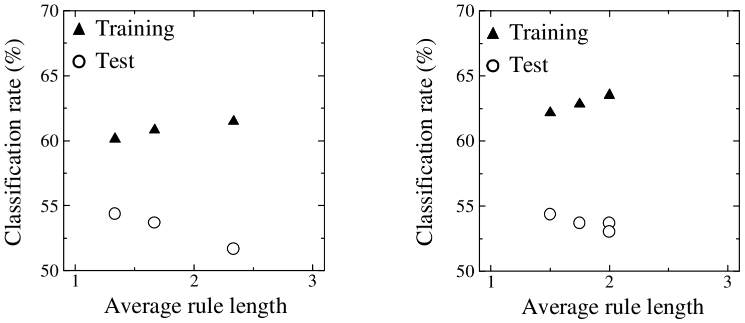
Fig. 18 Fuzzy rule-based classifier B with the highest training data accuracy in Figure 16 (Glass)

more circles (i.e., test data accuracy) than triangles (i.e., training data accuracy). This is because our multiobjective fuzzy rule selection algorithm found multiple fuzzy rule-based classifiers with the same training data accuracy and the same complexity (i.e., the same number of fuzzy rules and the same average rule length). Those fuzzy rule-based classifiers do not always have the same test data accuracy. Thus the number of circles (i.e., results for test data) is larger than the number of triangles (i.e., results for the training data) in Figure 20 (b).



(a) Training data accuracy. (b) Test data accuracy.

Fig. 19 Obtained non-dominated fuzzy rule-based classifiers (Heart C)

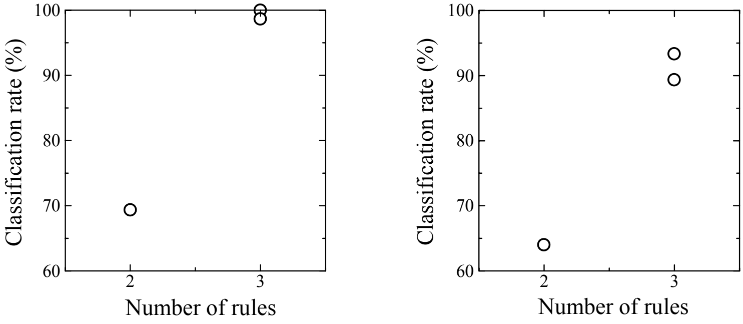


(a) Classifiers with three rules. (b) Classifiers with four rules.

Fig. 20 Relation between the accuracy and the average rule length (Heart C)

Iris Data Set: Experimental results are shown in Figure 21. All training patterns were correctly classified by three fuzzy rules in Figure 21 (a). Since a small fuzzy rule-based classifier had a 100% training data accuracy, many non-dominated fuzzy rule-based classifiers were not obtained. There exists, however, a clear accuracy-complexity tradeoff relation for training data in Figure 21 (a).

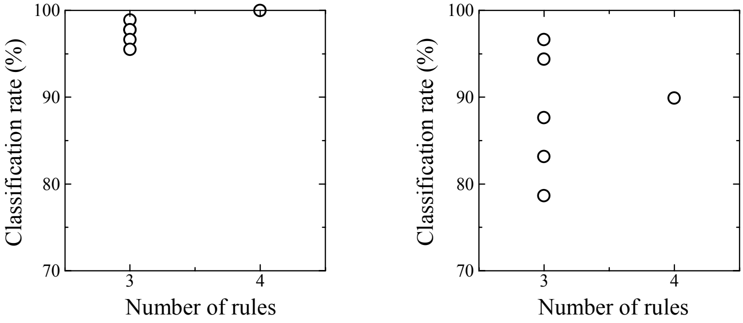
Wine Data Set: Experimental results are shown in Figure 22. All the training patterns were correctly classified by four fuzzy rules in Figure 22 (a). As in the case of the iris data set, a small fuzzy rule-based classifier had a 100% training data accuracy in Figure 22 (a). As a result, many non-dominated fuzzy rule-based classifiers were not obtained for the wine data set. Nevertheless we can observe an accuracy-complexity tradeoff relation for training data in Figure 22 (a). The highest test data accuracy was obtained by a fuzzy rule-based classifier with three fuzzy rules in Figure 22 (b). That fuzzy rule-based classifier also has the highest training data accuracy among the obtained fuzzy rule-based classifiers with three fuzzy rules (see Figure 23).



(a) Training data accuracy.

(b) Test data accuracy.

Fig. 21 Obtained non-dominated fuzzy rule-based classifiers (Iris)

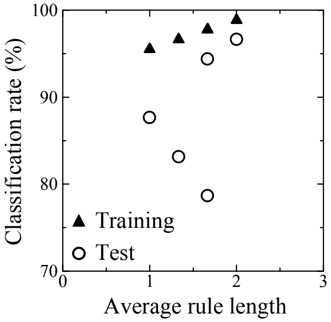


(a) Training data accuracy.

(b) Test data accuracy.

Fig. 22 Obtained non-dominated fuzzy rule-based classifiers (Wine)

Fig. 23 Relation between the accuracy and the average rule length of the obtained fuzzy rule-based classifiers with three rules (Wine)



In our computational experiments in this subsection, we always observed a clear tradeoff relation between the training data accuracy of fuzzy rule-based classifiers and the number of fuzzy rules. Such an accuracy-complexity tradeoff relation was not always clear for test data. For some data sets, we observed a clear tradeoff relation between the test data accuracy of fuzzy rule-based classifiers and the number of fuzzy rules. For other data sets, we observed the overfitting of fuzzy rule-based classifiers to training data due to the increase in the number of fuzzy rules. It should be noted that we can obtain these observations from a single run of NSGA-II for each data set. This is because NSGA-II (EMO algorithms in general) can search for a large number of non-dominated fuzzy rule-based classifiers by its single run.

5 Multiobjective Fuzzy Genetics-Based Machine Learning

In the previous section, NSGA-II was used for multiobjective fuzzy rule selection. Various approaches to the multiobjective design of fuzzy rule-based systems have also been proposed in the framework of genetics-based machine learning (GBML) where fuzzy rules are generated by genetic operations. In this section, we explain how NSGA-II can be used as a multiobjective fuzzy GBML algorithm.

5.1 Two Approaches in Genetics-Based Machine Learning

Applications of genetic algorithms to machine learning are referred to as genetics-based machine learning (GBML). Studies on GBML are often divided into two classes: Pittsburgh and Michigan approaches (for example, see [18, 48, 53] for GBML and [36, 37, 63, 80] for fuzzy GBML). A rule set is handled as an individual in the Pittsburgh approach while a single rule is handled as an individual in the Michigan approach. The final result (i.e., the finally obtained rule set) is usually the best individual in the final population in the Pittsburgh approach while it is the final population in the Michigan approach. Another category of GBML is an iterative rule learning approach [25, 35, 55] where a single rule is obtained from its single execution. Thus multiple runs are required to generate a rule set in the iterative rule learning approach. Multiobjective GBML algorithms have usually been implemented in the framework of the Pittsburgh approach. In many studies, the antecedent part of each rule is coded as a substring of integers and/or real numbers in Pittsburgh-style GBML algorithms. A rule set is represented by a concatenated string of multiple substrings.

A three-objective fuzzy GBML algorithm was compared with its rule selection version in [79]. A Pittsburgh-Michigan hybrid fuzzy GBML algorithm [91] was generalized as a multiobjective algorithm for interpretability-accuracy tradeoff analysis in [85]. Other examples of multiobjective fuzzy GBML algorithms are found in [10, 23, 24, 29, 54, 57, 80, 94, 97, 105, 124, 134, 135, 138, 140, 141, 144] where various aspects of fuzzy rule-based systems are adjusted by EMO algorithms (e.g., input selection, membership function tuning, and rule selection). Multiobjective GBML algorithms were also implemented for non-fuzzy classifier design (e.g., [110]).

5.2 Implementation of Multiobjective Fuzzy GBML Algorithms

In this subsection, we explain how a multiobjective fuzzy GBML algorithm can be implemented for efficiently finding non-dominated fuzzy rule-based classifiers in the same situation as in the previous section. That is, we assume that we have

m training patterns $\mathbf{x}_p = (x_{p1}, ..., x_{pn})$, $p = 1, 2, ..., m$ from M classes in an n-dimensional unit-hypercube $[0, 1]^n$. We use the 14 antecedent fuzzy sets in Figure 9 and *don't care* (i.e., the unit interval [0, 1]) in the antecedent part of fuzzy rules of the same form (i.e., fuzzy rules of the form in (11)). The antecedent part of each fuzzy rule is generated by genetic operations while its consequent class and rule weight are heuristically specified. We use the single winner-based fuzzy reasoning scheme for classifying each training patterns. We also use the same three objectives (i.e., $f_1(S)$: the number of correctly classified training patterns, $f_2(S)$: the number of selected fuzzy rules, and $f_3(S)$: the total number of antecedent conditions). Our multiobjective fuzzy GBML algorithm is implemented in the framework of NSGA-II. That is, each individual (i.e., fuzzy rule-based classifier) is evaluated by the non-dominated sorting and the crowding distance. The next population is generated from the current and offspring populations by the $(\mu + \lambda)$-ES generation update scheme where $\mu = \lambda$.

In our multiobjective fuzzy GBML algorithm, each fuzzy rule R_q is represented by its antecedent fuzzy sets A_{qi} ($i = 1, 2, ..., n$) as an integer string of length n where n is the dimensionality of the pattern space (i.e., n is the number of attributes of each pattern). We use 15 symbols (e.g., 0, 1, ..., 9, a, b, c, d, e) to represent *don't care* and the 14 antecedent fuzzy sets as shown in Figure 9. For example, an integer string "0102d0" denotes the fuzzy rule "If x_2 is S^2 and x_4 is L^2 and x_5 is ML^5 then Class C_q with CF_q" where *don't care* conditions on x_1, x_3 and x_6 represented by 0s in the string are omitted. It should be noted that the consequent class C_q and the rule weight CF_q are heuristically specified by compatible training patterns in the same manner as in the previous section.

A rule set S is handled as an individual and coded as a concatenated integer string where each substring of length n represents a single fuzzy rule. It should be noted that the number of fuzzy rules in each rule set is not fixed in our multiobjective fuzzy GBML algorithm. This means that we use integer strings of variable length as individuals.

It was shown in [77, 78] that the search ability of Michigan-style fuzzy GBML algorithms was drastically improved by directly generating initial fuzzy rules from training patterns in a heuristic manner. We use a similar heuristic method to generate an initial population of rule sets. First we randomly select a prespecified number of training patterns (say, N_{rule} training patterns where N_{rule} is the number of fuzzy rules in each initial rule set). Next we generate a fuzzy rule R_q from each training pattern $\mathbf{x}_p = (x_{p1}, ..., x_{pn})$ by probabilistically choosing an antecedent fuzzy set A_{qi} for each attribute value x_{pi} from the 14 antecedent fuzzy sets B_k ($k = 1, 2, ..., 9$, a, b, c, d, e) in Figure 9. Each antecedent fuzzy set B_k has the following selection probability for the attribute value x_{pi}:

$$P(B_k) = \frac{\mu_{B_k}(x_{pi})}{\sum_{j=1}^{\mathrm{e}} \mu_{B_j}(x_{pi})}, \quad k = 1, 2, ..., 9, \mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}. \tag{19}$$

That is, each antecedent fuzzy set B_k has a selection probability which is proportional to its compatibility grade with the attribute value x_{pi}. Then each antecedent fuzzy set of the generated fuzzy rule is replaced with *don′t care* using a prespecified probability $P_{don't\ care}$. In this manner, N_{rule} initial fuzzy rules are generated. An initial rule set consists of these fuzzy rules. By iterating this procedure, we generate N_{pop} initial rule sets (i.e., an initial population).

Two individuals (i.e., two rule sets) are selected from the current population by binary tournament selection with replacement in the same manner as in NSGA-II. Let the selected rule sets be S_1 and S_2. Some fuzzy rules are randomly selected from each parent to construct a new rule set by crossover. The number of fuzzy rules to be inherited from each parent to the new rule set is randomly specified. Let N_1 and N_2 be the number of fuzzy rules to be inherited from S_1 and S_2, respectively. We randomly specify N_1 and N_2 in the intervals $[1, |S_1|]$ and $[1, |S_2|]$, respectively, where $|S_i|$ is the number of fuzzy rules in the rule set S_i. In order to generate a new fuzzy rule, N_1 and N_2 fuzzy rules are randomly chosen from S_1 and S_2, respectively. The offspring rule set has $(N_1 + N_2)$ fuzzy rules. We use an upper bound on the number of fuzzy rules in each rule set (e.g., 40 in our computational experiments). When the number of fuzzy rules is larger than the upper bound, we randomly remove fuzzy rules from the offspring rule set until the upper bound condition is satisfied. The above-mentioned crossover operation is applied to the selected pair of parent rule sets with a prespecified crossover probability P_{C}. When the crossover operation is not applied, one of the two parent rule sets is randomly chosen as their offspring rule set. Each antecedent fuzzy set of fuzzy rules in the offspring rule set is randomly replaced with a different antecedent fuzzy set by mutation. The mutation operation is applied to each antecedent fuzzy set with a prespecified mutation probability P_{M}.

After the crossover and mutation operations, a single iteration of the following Michigan-style algorithm is applied to the newly generated offspring rule set S:

Step 1: Classify each training pattern by the rule set S. The fitness value of each fuzzy rule in S is the number of correctly classified training patterns by that rule.

Step 2: Generate N_{replace} fuzzy rules from the existing rules in S by genetic operations and from misclassified and/or rejected training patterns by the above-mentioned heuristic manner.

Step 3: Replace the worst N_{replace} fuzzy rules in S with the newly generated N_{replace} fuzzy rules.

In Step 2, N_{replace} fuzzy rules are to be generated. We generate at least a half of new fuzzy rules (i.e., at least $N_{\text{replace}}/2$ fuzzy rules) by genetic operations from the existing fuzzy rules in S. The probabilistic specification of each antecedent fuzzy set by (19) and the replacement with *don′t care* using the probability $P_{don't\ care}$ are used to generate the other fuzzy rules.

Let N_{MR} be the sum of the number of misclassified and rejected training patterns by the rule set S. When N_{MR} is less than or equal to $N_{\text{replace}}/2$, all the N_{MR} training patterns are used to generate new fuzzy rules. In this case, N_{MR} fuzzy rules are generated from the N_{MR} training patterns. The other fuzzy rules (i.e., $(N_{\text{replace}} - N_{\text{MR}})$ fuzzy rules) are generated by genetic operations. On the other hand, when N_{MR} is larger than $N_{\text{replace}}/2$, $N_{\text{replace}}/2$ patterns are randomly chosen from the N_{MR} misclassified or rejected training patterns. Then $N_{\text{replace}}/2$ fuzzy rules are directly generated from the chosen patterns. The other $N_{\text{replace}}/2$ fuzzy rules are generated by genetic operations.

When we generate a new fuzzy rule from existing rules in S by genetic operations, first a pair of parent fuzzy rules is selected from S using binary tournament selection with replacement. Then the standard uniform crossover operation is applied to the selected pair to generate a new fuzzy rule. Finally each antecedent fuzzy set is randomly replaced with a different one using a prespecified mutation probability. This procedure is iterated to generate a required number of new fuzzy rules (i.e., N_{replace} fuzzy rules including directly generated fuzzy rules from training patterns).

The number of replaced fuzzy rules (i.e., N_{replace}) is specified as $\lceil 0.2 \times |S| \rceil$ for each rule set S where $\lceil 0.2 \times |S| \rceil$ is the minimum integer not smaller than $0.2 \times |S|$. For example, one fuzzy rule is replaced when the number of fuzzy rules in S is less than or equal to five. In this case, the heuristic rule generation procedure from training patterns and the genetic operation-based procedure from existing rules are randomly evoked with the same probability when at least one training pattern is misclassified or rejected by the rule set S.

As we have already explained, a new rule set S is generated in our multiobjective fuzzy GBML algorithm by selection, crossover, mutation, and a single iteration of the Michigan-style algorithm. Whereas unnecessary fuzzy rules were removed from each rule set in the multiobjective genetic fuzzy rule selection algorithm in the previous section, they are not removed in our multiobjective fuzzy GBML algorithm in this section. This is because unnecessary fuzzy rules may include useful antecedent fuzzy sets. Effects of the removal of unnecessary fuzzy rules on the performance of our multiobjective fuzzy GBML algorithm, however, should be examined in detail in future studies. The above-mentioned rule set generation procedure is iterated N_{pop} times to generate an offspring population of N_{pop} rule sets. The next population is constructed by choosing the best N_{pop} rule sets from the current and offspring populations in the same manner as in NSGA-II. Generation update is iterated until a prespecified stopping condition is satisfied. The total number of generations was used as the stopping condition in our computational experiments in this section.

5.3 Computational Experiments

We applied our multiobjective fuzzy GBML algorithm to the five data sets in Table 1 using the following parameter specifications:

Number of fuzzy rules in each initial rule set: 20 rules,
Probability of *don't care* ($P_{don't\ care}$): 0.8,
Population size: 200 rule sets,
Crossover probability in the Pittsburgh-style part: 0.9,
Crossover probability in the Michigan-style part: 0.9,

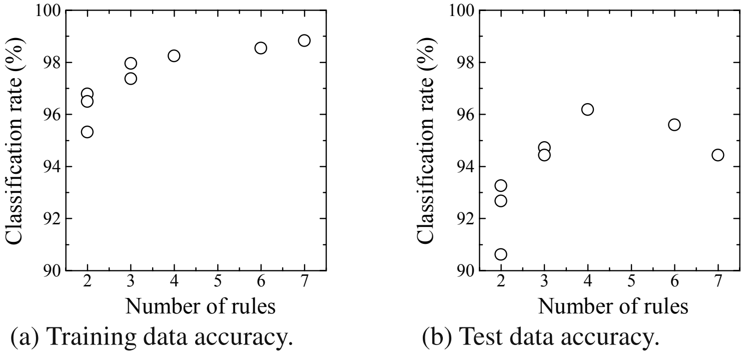


Fig. 24 Obtained non-dominated fuzzy rule-based classifiers (Breast W)

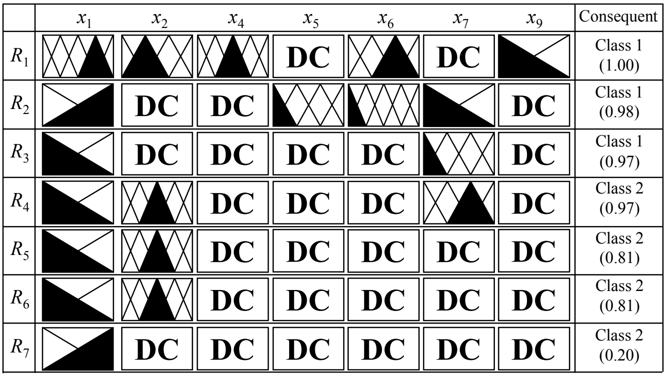
Mutation probability in the Pittsburgh-style part: $1/n$,
Mutation probability in the Michigan-style part: $1/n$,
Stopping condition: 5000 generations.

We used the same population size and the same stopping condition as in the previous section for multiobjective fuzzy rule selection. We also used the same partition of each data set into training patterns and test patterns. Experimental results by our multiobjective fuzzy GBML algorithm are briefly reported in the following.

Wisconsin Breast Cancer Data Set: Experimental results are summarized in Figure 24. Our multiobjective fuzzy GBML algorithm found 10 non-dominated rule sets. From the comparison between Figure 10 (a) and Figure 24 (a), we can see that very similar results were obtained for training data. We can also see that the most complicated fuzzy rule-based classifier with the highest training data accuracy in Figure 24 (a) does not have the highest test data accuracy in Figure 24 (b). This classifier, which is shown in Figure 25, includes long fuzzy rules with many antecedent conditions as well as short ones. Computation time for multiobjective fuzzy GBML was 180 minutes on a PC with Intel Xeon 3.6GHz with 4GB RAM in Figure 24 while it was 17 minutes for multiobjective fuzzy rule selection including candidate rule generation in Figure 10. This difference is due to a tailored efficient implementation of multiobjective fuzzy rule selection where the compatibility grade of each candidate rule to each training pattern was calculated just once and stored during the execution of NSGA-II. Such an efficient implementation of multiobjective fuzzy GBML is a future research issue.

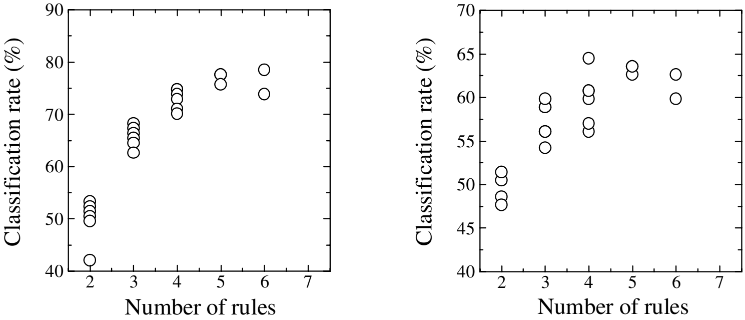
Glass Data Set: Experimental results are shown in Figure 26. Our multiobjective fuzzy GBML algorithm found 28 non-dominated fuzzy rule-based classifiers. Similar results were obtained in Figure 16 and Figure 26 except for complicated classifiers with more than six rules (i.e., those classifiers were not obtained in Figure 26).

Cleveland Heart Disease Data Set: Experimental results are shown in Figure 27. Our multiobjective fuzzy GBML algorithm found 21 non-dominated fuzzy rule-based classifiers. As in Figure 26, complicated fuzzy rule-based classifiers were not



	x_1	x_2	x_4	x_5	x_6	x_7	x_9	Consequent
R_1				DC		DC		Class 1 (1.00)
R_2		DC	DC				DC	Class 1 (0.98)
R_3		DC	DC	DC	DC		DC	Class 1 (0.97)
R_4			DC	DC	DC		DC	Class 2 (0.97)
R_5			DC	DC	DC	DC	DC	Class 2 (0.81)
R_6			DC	DC	DC	DC	DC	Class 2 (0.81)
R_7		DC	DC	DC	DC	DC	DC	Class 2 (0.20)

Fig. 25 The most complicated fuzzy rule-based classifier with the highest training data accuracy in Figure 24 (Breast W). The fifth and sixth fuzzy rules are exactly the same. We can remove one of them without changing any classification results. Such a duplicated fuzzy rule is removed as an unnecessary rule if we include the unnecessary rule removal mechanism into our fuzzy GBML algorithm



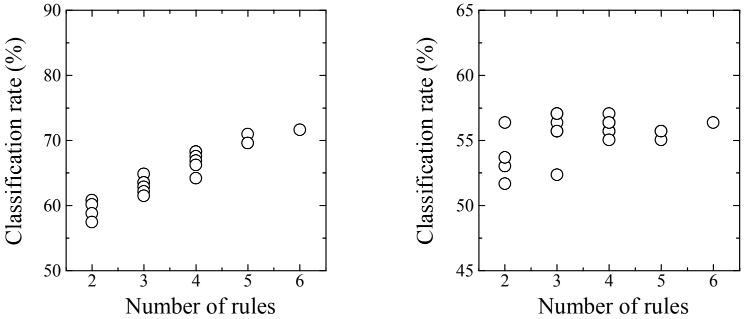
(a) Training data accuracy. (b) Test data accuracy.

Fig. 26 Obtained non-dominated fuzzy rule-based classifiers (Glass)

obtained in Figure 27 (compare Figure 27 with Figure 19). Similar results, however, were achieved in Figure 27 (b) and Figure 19 (b) with respect to classification rates on test data.

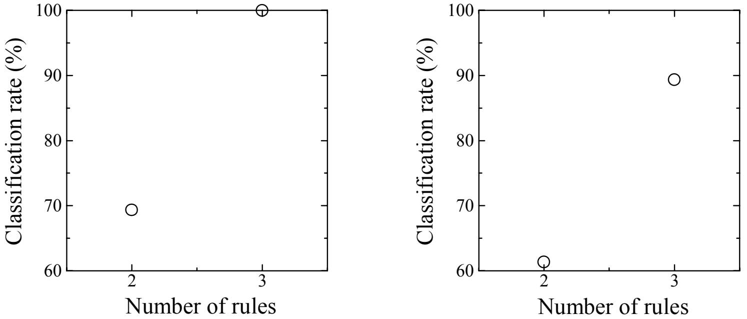
Iris Data Set: Experimental results are shown in Figure 28. As in Figure 21 (a), all training patterns were correctly classified by three fuzzy rules in Figure 28 (a).

Wine Data Set: Experimental results are shown in Figure 29. All training patterns were correctly classified by five fuzzy rules in Figure 29 (a) while they were correctly classified by four fuzzy rules in Figure 22 (a). That is, better results were obtained by rule selection in Figure 22 (a) than GBML in Figure 29 (a) for training data. On the other hand, similar performance was obtained by these two approaches with respect to classification rates on test patterns in Figure 22 (b) and Figure 29 (b).



(a) Training data accuracy. (b) Test data accuracy.

Fig. 27 Obtained non-dominated fuzzy rule-based classifiers (Heart C)



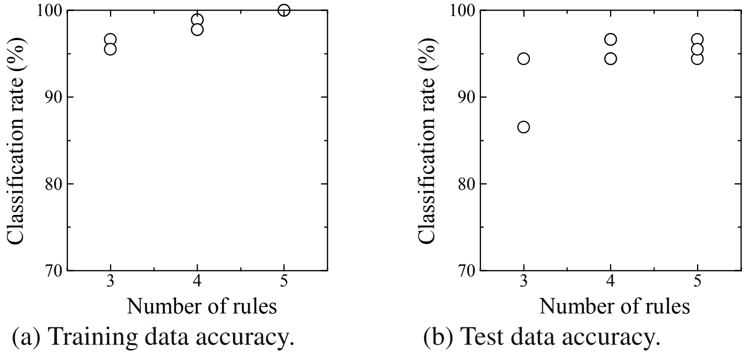
(a) Training data accuracy. (b) Test data accuracy.

Fig. 28 Obtained non-dominated fuzzy rule-based classifiers (Iris)

We did not observe any clear differences between experimental results by the two approaches: Multiobjective fuzzy rule selection in the previous section and multiobjective fuzzy GBML in this section. In Figure 30, we compare these two approaches in terms of the average rule length. We can see from Figure 30 that much longer fuzzy rules were included in fuzzy rule-based classifiers obtained by GBML than rule selection. This is because only short fuzzy rules were used as candidate rules in rule selection.

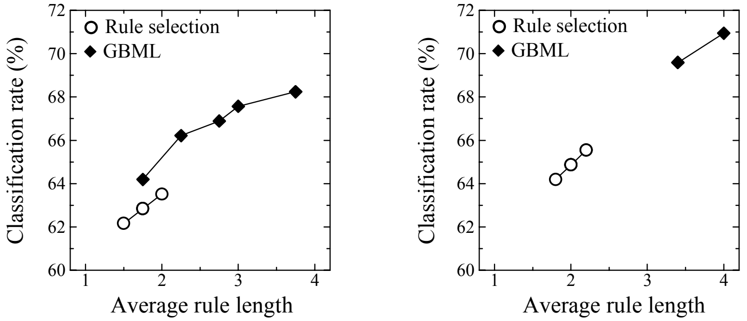
Since multiobjective fuzzy GBML can generate any fuzzy rule with an arbitrary number of antecedent conditions, fuzzy rule-based classifiers may include long fuzzy rules as well as short ones (e.g., see Figure 25). This means that GBML has a much larger search space than rule selection [85]. Thus GBML may need much more computation load (i.e., a larger population size and/or a larger number of generations) whereas we used the same specification for these two approaches.

Since the performance and the computation time of rule selection strongly depend on the choice of candidate rules, the above-mentioned observations with respect to the comparison between the two algorithms are not always valid. Different observations may be obtained when we use different specifications of candidate



(a) Training data accuracy. (b) Test data accuracy.

Fig. 29 Obtained non-dominated fuzzy rule-based classifiers (Wine)



(a) Classifiers with four rules. (b) Classifier with five rules.

Fig. 30 Relation between the accuracy and the average rule length (Heart C)

rules in multiobjective fuzzy rule selection. Multiobjective fuzzy GBML can be implemented in a more efficient manner by including some heuristics (e.g., an upper bound on rule length, unnecessary rule removal, and rule removal mutation) as in multiobjective fuzzy rule selection.

6 Related Studies

In this section, we briefly explain various studies related to multiobjective genetic fuzzy systems. More detailed explanations can be found in [97] for multiobjective machine learning and [51] for multiobjective data mining.

6.1 Evolutionary Multiobjective Data Mining

Evolutionary algorithms have been applied to knowledge discovery and data mining in various manners [48]. Recently EMO algorithms have been used for two different tasks: One is to search for Pareto-optimal rules and the other is to search for Pareto-optimal rule sets.

In data mining techniques such as Apriori [7], *support* and *confidence* have frequently been used as rule evaluation criteria. Other rule evaluation criteria, however, were also proposed. Among them are *gain*, *variance*, *chi-squared value*, *entropy gain*, *gini*, *laplace*, *lift*, and *conviction* [15]. It was shown for non-fuzzy rules that the best rule according to any of these measures is a Pareto-optimal rule of the following two-objective rule discovery problem [15]:

$$\text{Maximize } \{\mathit{Support}(R),\ \mathit{Confidence}(R)\}, \tag{20}$$

where R denotes a single rule.

The use of NSGA-II [42] was proposed in [71, 73] to search for Pareto-optimal classification rules of the two-objective data mining problem in (20). A dissimilarity measure between classification rules was used in [72] instead of the crowding distance in NSGA-II to increase the diversity of obtained Pareto-optimal rules. The Pareto-dominance relation used in NSGA-II was modified in [126] in order to search for not only Pareto-optimal classification rules but also near Pareto-optimal ones. Similar multiobjective formulations to (20) were used to search for Pareto-optimal association rules [52] and Pareto-optimal fuzzy association rules [103]. In [65], the tradeoff between the number of extracted fuzzy rules and the computation time for rule extraction was discussed in fuzzy data mining.

The above-mentioned studies on multiobjective fuzzy rule selection and multiobjective fuzzy GBML in the previous sections can be viewed as data mining techniques for finding Pareto-optimal rule sets. In [74], the use of Pareto-optimal fuzzy rules as candidate rules was examined in multiobjective fuzzy rule selection.

6.2 Evolutionary Multiobjective Feature Selection

Feature selection [109] is an important issue in modeling, classification, knowledge discovery and data mining. The basic idea of multiobjective feature selection is to minimize the size of a subset of features and maximize its performance. There exists a clear tradeoff relation between the size of feature subsets and their performance on training data. Evolutionary multiobjective feature selection was discussed in some studies (e.g., [45, 120, 121]). Feature selection was also discussed in the context of multiobjective genetic fuzzy systems [39].

6.3 Evolutionary Multiobjective Clustering

Fuzzy clustering [16] has frequently been used for fuzzy rule generation. In evolutionary multiobjective clustering [59, 60, 61, 62], multiple cluster quality measures are optimized simultaneously. Evolutionary multiobjective clustering will play a very important role in multiobjective design of fuzzy rule-based systems whereas it has not been used in many studies. The number of clusters can be used as an objective function to be minimized in multiobjective clustering since it is directly related to the number of fuzzy rules in the design of fuzzy rule-based systems.

6.4 Evolutionary Ensemble Design

A promising approach to the design of reliable classifiers is to combine multiple classifiers into a single one [13, 44]. Several methods have been proposed for generating multiple classifiers such as bagging [19] and boosting [49]. The point in the design of a high-performance ensemble classifier is to generate an ensemble of classifiers with high diversity. Ideally the classification errors by each individual classifier in an ensemble should be uncorrelated.

EMO algorithms have been used to generate an ensemble of classifiers with high diversity. Non-dominated neural networks were combined into a single ensemble classifier in [1, 26, 27, 98]. The choice of ensemble members seems to be an interesting issue when a large number of non-dominated neural networks are obtained. Design of fuzzy ensemble classifiers was discussed in [82, 88, 117]. Feature selection was used for neural network ensemble design in [119, 122].

6.5 Evolutionary Multiobjective Neural Network Design

In addition to ensemble design, EMO algorithms have also been used for multiobjective design of neural networks in various manners. An EMO algorithm was used to generate a number of non-dominated neural networks on a receiver operating characteristic curve in [107]. Non-dominated radial basis function (RBF) networks of different sizes were generated in [56]. A multiobjective memetic algorithm was used to speed up the back-propagation algorithm in [2] where a number of neural networks of different sizes were evolved to find an appropriate network structure.

6.6 Multiobjective Genetic Programming

As in fuzzy rule-based systems and neural networks, there exists a clear tradeoff relation between the training data accuracy and the size of trees in genetic programming. Multiobjective genetic programming has been discussed in many studies [11, 17, 38, 127, 129]. In standard single-objective genetic programming, constraints and/or penalties with respect to the size of trees have often been used to prevent too complicated trees. Multiobjective genetic programming seems to be a promising approach to the multiobjective design of tree-structured fuzzy systems. Multiobjective genetic programming can be used to search for various trees with different tradeoffs between the structural complexity and the training data accuracy.

7 Future Research Directions

An important future research issue is the formulation of interpretability of fuzzy rule-based systems as complexity measures. Various aspects are related to their interpretability (e.g., the number of input variables, the number of fuzzy sets for each variable, the separability of adjacent fuzzy sets, the number of fuzzy rules, the number of antecedent conditions of each fuzzy rule, etc.). See [23, 24, 58, 83, 113] for further discussions on interpretability of fuzzy rule-based systems. If we use those aspects as separate objectives, fuzzy system design is formulated as a many-objective problem. Pareto ranking-based EMO algorithms such as NSGA-II [42] and SPEA [147], however, do not work well on such a problem with many objectives [70, 87, 93, 104, 125]. Thus it is necessary to choose only a few interpretability measures or to combine various interpretability measures into a few objective functions. It would be interesting to examine how the search ability of EMO algorithms for multiobjective fuzzy system design depends on the choice of interpretability measures.

Performance evaluation of multiobjective genetic fuzzy systems is also very important. This issue is two-fold. One is related to the performance of a finally selected fuzzy rule-based system. After a single fuzzy rule-based system is chosen from a large number of obtained non-dominated ones, its generalization ability together with its interpretability can be compared with results by single-objective approaches. Such a comparative study may clearly demonstrate advantages and disadvantages of multiobjective approaches over single-objective ones. Another performance evaluation issue is related to the search ability of multiobjective genetic fuzzy systems as multiobjective optimizers. A number of performance indices for evaluating EMO algorithms [32, 34, 41, 47, 148] can be used for this task. It should be noted that the search ability of EMO algorithms in multiobjective genetic fuzzy systems is evaluated by training data accuracy (i.e., accuracy measures in multiobjective problems) while the performance of obtained fuzzy rule-based systems is evaluated by test data accuracy (i.e., actual performance of fuzzy rule-based systems).

Another future research issue is theoretical analysis for maximizing the generalization ability of fuzzy rule-based systems. As shown in this chapter, multiobjective genetic fuzzy systems can be used for empirical analysis of accuracy-complexity tradeoff of fuzzy rule-based systems [85]. Almost all studies on multiobjective genetic fuzzy systems are based on computational experiments with no theoretical analysis. Theoretical analysis such as statistical learning theory [30] seems to be an important research issue for finding fuzzy rule-based systems with high generalization ability. In this context, regularization methods can be discussed as multiobjective problems [99].

Incorporation of user's preference is a hot issue in the EMO community [5, 31, 40, 43, 84, 96]. User's preference can be incorporated into multiobjective genetic fuzzy systems in order to efficiently search for preferred fuzzy rule-based systems. Some users may prefer accurate fuzzy rule-based systems even if its interpretability is not high. Other users may prefer simple fuzzy rule-based systems even if its accuracy is not very high. Interpretability is important in some application areas while accuracy is the primary objective in many studies on the design of fuzzy rule-based systems. Information on user's preference can be used to guide the multiobjective search for preferred fuzzy rule-based systems. The choice of a single fuzzy rule-based system from a large number of obtained non-dominated ones should be also discussed in future studies. In this context, visualization of obtained non-dominated fuzzy rule-based systems seems to play an important role.

As in genetic fuzzy systems [36, 37, 63], genetic algorithms have been dominantly used in multiobjective genetic fuzzy systems. The use of other techniques such as multiobjective particle swarm optimization [6, 14, 33, 64] and multiobjective differential evolution [12, 28, 108, 128] will be examined for multiobjective design of fuzzy rule-based systems. The use of multiobjective clustering and multiobjective genetic programming will be also examined in the near future.

Finally we need efficient tricks for the handling of large data sets in evolutionary algorithms (e.g., stratification [20]). Parallel implementation of evolutionary computation [8, 21, 114] seems to be a promising research direction in genetic fuzzy systems (e.g., parallel implementation with data set stratification [118]).

8 Conclusions

Linguistic interpretability is the main advantage of fuzzy rule-based systems over other nonlinear models such as neural networks. Accuracy maximization, however, often leads to the deterioration in the linguistic interpretability (i.e., increase in the complexity) of fuzzy rule-based systems. As a promising approach to the handling of the tradeoff between accuracy maximization and complexity minimization, we explained multiobjective design of fuzzy rule-based systems in this chapter. A large number of non-dominated fuzzy rule-based systems with respect to accuracy maximization and complexity minimization can be obtained by a single run of an EMO algorithm. Through computational experiments, we demonstrated that an accuracy-complexity tradeoff relation can be visually shown by the obtained non-dominated fuzzy rule-based systems. Human users are supposed to choose a single fuzzy rule-based system from the obtained non-dominated ones by their preference with respect to accuracy and interpretability. In this chapter, we also briefly explained a wide range of related studies to evolutionary multiobjective design of fuzzy rule-based systems. Moreover, we pointed out some future research directions in the field of multiobjective genetic fuzzy systems.

References

1. Abbass, H.A.: Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization. In: Proc. of CEC 2003, pp. 2074–2080 (2003)
2. Abbass, H.A.: Speeding up back-propagation using multiobjective evolutionary algorithms. Neural Comput. 15, 2705–2726 (2003)
3. Abe, S.: Pattern classification: Neuro-fuzzy methods and their comparison. Springer, London (2001)
4. Abonyi, J., Roubos, J.A., Szeifert, F.: Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision-tree initialization. Int. J. of Approx. Reason 32, 1–21 (2003)

5. Adra, S.F., Griffin, I., Fleming, P.J.: A comparative study of progressive preference articulation techniques for multiobjective optimisation. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 908–921. Springer, Heidelberg (2007)
6. Agrawal, S., Dashora, Y., Tiwari, M.K., Son, Y.J.: Interactive particle swarm: A Pareto-adaptive metaheuristic to multiobjective optimization. IEEE Trans. on Syst. Man and Cybern. - Part A 38, 258–277 (2008)
7. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 307–328. AAAI, Menlo Park (1996)
8. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Trans. on Evol. Comput. 6, 443–462 (2002)
9. Alcala, R., Cano, J.R., Cordon, O., Herrera, F., Villar, P., Zwir, I.: Linguistic modeling with hierarchical systems of weighted linguistic rules. Int. J. of Approx. Reason 32, 187–215 (2003)
10. Alcala, R., Alcala-Fdez, J., Gacto, M.J., Herrera, F.: A multi-objective genetic algorithm for tuning and rule selection to obtain accurate and compact linguistic fuzzy rule-based systems. Int. J. of Uncertain Fuzziness and Knowl-Based Syst. 15, 539–557 (2007)
11. Araujo, L.: Multiobjective genetic programming for natural language parsing and tagging. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 433–442. Springer, Heidelberg (2006)
12. Babu, B.V., Chakole, P.G., Mubeen, J.H.S.: Multiobjective differential evolution (MODE) for optimization of adiabatic styrene reactor. Chem. Eng. Sci. 60, 4822–4837 (2005)
13. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Mach. Learn. 36, 105–139 (1999)
14. Baumgartner, U., Magele, C., Renhart, W.: Pareto optimality and particle swarm optimization. IEEE Trans. on Magn. 40, 1172–1175 (2004)
15. Bayardo Jr., R.J., Agrawal, R.: Mining the most interesting rules. In: Proc. of KDD, pp. 145–153 (1999)
16. Bezdek, J.C.: Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York (1981)
17. Bleuler, S., Brack, M., Thiele, L., Zitzler, E.: Multiobjective genetic programming: Reducing bloat using SPEA2. In: Proc. of CEC 2001, pp. 536–543 (2001)
18. Booker, L.B., Goldberg, D.E., Holland, J.H.: Classifier systems and genetic algorithms. Artif. Intell. 40, 235–282 (1989)
19. Breiman, L.: Bagging predictors. Mach. Learn. 24, 123–140 (1996)
20. Cano, J.R., Herrera, F., Lozano, M.: Stratification for scaling up evolutionary prototype selection. Pattern Recognit. Lett. 26, 953–963 (2005)
21. Cantu-Paz, E.: Efficient and accurate parallel genetic algorithms. Springer, Berlin (2000)
22. Casillas, J., Cordon, O., del Jesus, M.J., Herrera, F.: Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction. IEEE Trans. on Fuzzy Syst. 13, 13–29 (2005)
23. Casillas, J., Cordon, O., Herrera, F., Magdalena, L. (eds.): Accuracy improvements in linguistic fuzzy modeling. Springer, Berlin (2003)

24. Casillas, J., Cordon, O., Herrera, F., Magdalena, L. (eds.): Interpretability issues in fuzzy modeling. Springer, Berlin (2003)
25. Castillo, L., Gonzalez, A., Perez, R.: Including a simplicity criterion in the selection of the best rule in a genetic fuzzy learning algorithm. Fuzzy Sets and Syst. 120, 309–321 (2001)
26. Chandra, A., Yao, X.: DIVACE: Diverse and accurate ensemble learning algorithm. In: Yang, Z.R., Yin, H., Everson, R.M. (eds.) IDEAL 2004, vol. 3177, pp. 619–625. Springer, Heidelberg (2004)
27. Chandra, A., Yao, X.: Evolutionary framework for the construction of diverse hybrid ensemble. In: Proc. of ESANN 2005, pp. 253–258 (2005)
28. Chang, Y.P., Wu, C.J.: Optimal multiobjective planning of large-scale passive harmonic filters using hybrid differential evolution method considering parameter and loading uncertainty. IEEE Trans. on Power Deliv. 20, 408–416 (2005)
29. Chen, L.H., Chiang, C.H.: An intelligent control system with a multi-objective self-exploration process. Fuzzy Sets and Syst. 143, 275–294 (2004)
30. Cherkassky, V., Mulier, F.: Learning from data: Concepts, theory, and methods. John Wiley & Sons, New York (1998)
31. Coello, C.A.C.: Handling preferences in evolutionary multiobjective optimization: A survey. In: Proc. of CEC 2000, pp. 30–37 (2000)
32. Coello, C.A.C., Lamont, G.B.: Applications of multi-objective evolutionary algorithms. World Scientific, Singapore (2004)
33. Coello, C.A.C., Pulido, G.T., Lechuga, M.S.: Handling multiple objectives with particle swarm optimization. IEEE Trans. on Evol. Comput. 8, 256–279 (2004)
34. Coello, C.A.C., van Veldhuizen, D.A., Lamont, G.B.: Evolutionary algorithms for solving multi-objective problems. Kluwer Academic Publishers, Boston (2002)
35. Cordon, O., del Jesus, M.J., Herrera, F., Lozano, M.: MOGUL: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach. Int. J. of Intell. Syst. 14, 1123–1153 (1999)
36. Cordon, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L.: Ten years of genetic fuzzy systems: Current framework and new trends. Fuzzy Sets and Syst. 141, 5–31 (2004)
37. Cordon, O., Herrera, F., Hoffmann, F., Magdalena, L.: Genetic Fuzzy Systems. World Scientific, Singapore (2001)
38. Cordon, O., Herrera-Viedma, E., Luque, M.: Evolutionary learning of Boolean queries by multiobjective genetic programming. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 710–719. Springer, Heidelberg (2002)
39. Cordon, O., Jesus, M.J.D., Herrera, F., Magdalena, L., Villar, P.: A multiobjective genetic learning process for joint feature selection and granularity and contexts learning in fuzzy rule-based classification systems. In: Casillas, J., Cordon, O., Herrera, F., Magdalena, L. (eds.) Interpretability issues in fuzzy modeling, pp. 79–99. Springer, Berlin (2003)
40. Cvetkovic, D., Parmee, I.C.: Preferences and their application in evolutionary multiobjective optimization. IEEE Trans. on Evol. Comput. 6, 42–57 (2002)
41. Deb, K.: Multi-objective optimization using evolutionary algorithms. John Wiley & Sons, Chichester (2001)
42. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evol. Comput. 6, 182–197 (2002)

43. Deb, K., Sundar, J.: Reference point based multi-objective optimization using evolutionary algorithms. In: Proc. of GECCO 2006, pp. 635–642 (2006)
44. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Mach. Learn. 40, 139–157 (2000)
45. Emmanouilidis, C., Hunter, A., MacIntyre, J.: A multiobjective evolutionary setting for feature selection and acommonality-based crossover operator. In: Proc. of CEC 2000, pp. 309–316 (2000)
46. Feldman, D.S.: Fuzzy network synthesis with genetic algorithms. In: Proc. of 5th ICGA, pp. 312–317 (1995)
47. da Fonseca, V.G., Fonseca, C.M., Hall, A.O.: Inferential performance assessment of stochastic optimizers and the attainment function. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 213–225. Springer, Heidelberg (2001)
48. Freitas, A.: Data mining and knowledge discovery with evolutionary algorithms. Springer, Berlin (2002)
49. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. of Comput. and Syst. Sci. 55, 119–139 (1997)
50. Funahashi, K.: On the approximate realization of continuous-mappings by neural networks. Neural Netw. 2, 83–192 (1989)
51. Ghosh, A., Dehuri, S., Ghosh, S. (eds.): Multi-objective evolutionary algorithms for knowledge discovery from databases. Springer, Berlin (2008)
52. Ghosh, A., Nath, B.T.: Multi-objective rule mining using genetic algorithms. Inf. Sci. 163, 123–133 (2004)
53. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
54. Gomez-Skarmeta, A.F., Jimenez, F., Sanchez, G.: Improving interpretability in approximative fuzzy models via multiobjective evolutionary algorithms. Int. J. of Intell. Syst. 22, 943–969 (2007)
55. Gonzalez, A., Perez, R.: SLAVE: A genetic learning system based on an iterative approach. IEEE Trans. on Fuzzy Syst. 7, 176–191 (1999)
56. Gonzalez, J., Rojas, I., Ortega, J., Pomares, H., Fernandez, F.J., Diaz, A.F.: Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. IEEE Trans. on Neural Netw. 14, 1478–1495 (2003)
57. Gonzalez, J., Rojas, I., Pomares, H., Herrera, L.J., Guillen, A., Palomares, J.M., Rojas, F.: Improving the accuracy while preserving the interpretability of fuzzy function approximators by means of multi-objective evolutionary algorithms. Int. J. of Approx. Reason 44, 32–44 (2007)
58. Guillaume, S.: Designing fuzzy inference systems from data: An interpretability-oriented review. IEEE Trans. on Fuzzy Syst. 9, 426–443 (2001)
59. Handl, J., Knowles, J.: Evolutionary multiobjective clustering. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 1081–1091. Springer, Heidelberg (2004)
60. Handl, J., Knowles, J.: Multiobjective clustering around medoids. In: Proc. of CEC 2005, pp. 632–639 (2005)
61. Handl, J., Knowles, J.: Improving the scalability of multiobjective clustering. In: Proc. of CEC 2005, pp. 2372–2379 (2005)

62. Handl, J., Knowles, J.: Exploiting the trade-off - The benefits of multiple objectives in data clustering. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 547–560. Springer, Heidelberg (2005)
63. Herrera, F.: Genetic fuzzy systems: Status, critical considerations and future directions. Int. J. of Comput. Intell. Res. 1, 59–67 (2005)
64. Ho, S.L., Yang, S.Y., Ni, G.Z., Lo, E.W.C., Wong, H.C.: A particle swarm optimization-based method for multiobjective design optimizations. IEEE Trans. on Magn. 41, 1756–1759 (2005)
65. Hong, T.P., Kuo, C.S., Chi, S.C.: Trade-off between computation time and number of rules for fuzzy mining from quantitative data. Int. J. of Uncertain, Fuzziness and Knowl-Based Syst. 9, 587–604 (2001)
66. Horikawa, S., Furuhashi, T., Uchikawa, Y.: On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. IEEE Trans. on Neural Netw. 3, 801–806 (1993)
67. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. 2, 359–366 (1989)
68. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Netw. 4, 251–257 (1991)
69. Hu, Y.C., Chen, R.S., Tzeng, G.H.: Finding fuzzy classification rules using data mining techniques. Pattern Recognit. Lett. 24, 509–519 (2003)
70. Hughes, E.J.: Evolutionary many-objective optimization: many once or one many? In: Proc. of CEC 2005, pp. 222–227 (2005)
71. de la Iglesia, B., Philpott, M.S., Bagnall, A.J., Rayward-Smith, V.J.: Data mining rules using multi-objective evolutionary algorithms. In: Proc. of CEC 2003, pp. 1552–1559 (2003)
72. de la Iglesia, B., Reynolds, A., Rayward-Smith, V.J.: Developments on a multi-objective metaheuristic (MOMH) algorithm for finding interesting sets of classification rules. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 826–840. Springer, Heidelberg (2005)
73. de la Iglesia, B., Richards, G., Philpott, M.S., Rayward-Smith, V.J.: The application and effectiveness of a multi-objective metaheuristic algorithm for partial classification. Europ. J. of Oper. Res. 169, 898–917 (2006)
74. Ishibuchi, H., Kuwajima, I., Nojima, Y.: Relation between Pareto-optimal fuzzy rules and Pareto-optimal fuzzy rule sets. In: Proc of IEEE MCDM 2007, pp. 42–49 (2007)
75. Ishibuchi, H., Murata, T., Turksen, I.B.: Selecting linguistic classification rules by two-objective genetic algorithms. In: Proc. of SMC 1995, pp. 1410–1415 (1995)
76. Ishibuchi, H., Murata, T., Turksen, I.B.: Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. Fuzzy Sets and Syst. 89, 135–150 (1997)
77. Ishibuchi, H., Nakashima, T.: Improving the performance of fuzzy classifier systems for pattern classification problems with continuous attributes. IEEE Trans. on Ind. Electron 46, 157–168 (1999)
78. Ishibuchi, H., Nakashima, T., Murata, T.: Performance evaluation of fuzzy classifier systems for multi-dimensional pattern classification problems. IEEE Trans. on Syst. Man and Cybern. - Part B 29, 601–618 (1999)
79. Ishibuchi, H., Nakashima, T., Murata, T.: Three-objective genetics-based machine learning for linguistic rule extraction. Inf. Sci. 136, 109–133 (2001)
80. Ishibuchi, H., Nakashima, T., Nii, M.: Classification and modeling with linguistic information granules: Advanced approaches to linguistic data mining. Springer, Berlin (2004)

81. Ishibuchi, H., Namba, S.: Evolutionary multiobjective knowledge extraction for high-dimensional pattern classification problems. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 1123–1132. Springer, Heidelberg (2004)
82. Ishibuchi, H., Nojima, Y.: Evolutionary multiobjective optimization for the design of fuzzy rule-based ensemble classifiers. Int. J. of Hybrid Intell. Syst. 3, 129–145 (2006)
83. Ishibuchi, H., Nojima, Y., Kuwajima, I.: Finding simple fuzzy classification systems with high interpretability through multiobjective rule selection. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) KES 2006. LNCS, vol. 4252, pp. 86–93. Springer, Heidelberg (2006)
84. Ishibuchi, H., Nojima, Y.: Optimization of scalarizing functions through evolutionary multiobjective optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 51–65. Springer, Heidelberg (2007)
85. Ishibuchi, H., Nojima, Y.: Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. Int. J. of Approx. Reason 44, 4–31 (2007)
86. Ishibuchi, H., Nozaki, K., Yamamoto, N., Tanaka, H.: Selecting fuzzy if-then rules for classification problems using genetic algorithms. IEEE Trans. on Fuzzy Syst. 3, 260–270 (1995)
87. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary Many-Objective Optimization: A short Review. In: Proc. of CEC 2008, pp. 2424–2431 (2008)
88. Ishibuchi, H., Yamamoto, T.: Evolutionary multiobjective optimization for generating an ensemble of fuzzy rule-based classifiers. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 1077–1088. Springer, Heidelberg (2003)
89. Ishibuchi, H., Yamamoto, T.: Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. Fuzzy Sets and Syst. 141, 59–88 (2004)
90. Ishibuchi, H., Yamamoto, T.: Rule weight specification in fuzzy rule-based classification systems. IEEE Trans. on Fuzzy Syst. 13, 428–435 (2005)
91. Ishibuchi, H., Yamamoto, T., Nakashima, T.: Hybridization of fuzzy GBML approaches for pattern classification problems. IEEE Trans. on Syst. Man and Cybern. - Part B 35, 359–365 (2005)
92. Jang, J.S.R.: ANFIS: Adaptive-network-based fuzzy inference system. IEEE Trans. on Syst. Man and Cybern. 23, 665–685 (1993)
93. Jaszkiewicz, A.: On the computational efficiency of multiple objective metaheuristics: The knapsack problem case study. Europ. J. of Oper. Res. 158, 418–433 (2004)
94. Jiménez, F., Gómez-Skarmeta, A.F., Roubos, H., Babuska, R.: Accurate, transparent, and compact fuzzy models for function approximation and dynamic modeling through multi-objective evolutionary optimization. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 653–667. Springer, Heidelberg (2001)
95. Jin, Y.: Fuzzy modeling of high-dimensional systems: Complexity reduction and interpretability improvement. IEEE Trans. on Fuzzy Syst. 8, 212–221 (2000)
96. Jin, Y. (ed.): Knowledge incorporation in evolutionary computation. Springer, Berlin (2005)
97. Jin, Y. (ed.): Multi-objective machine learning. Springer, Berlin (2006)

98. Jin, Y., Okabe, T., Sendhoff, B.: Neural network regularization and ensembling using multi-objective evolutionary algorithms. In: Proc. of CEC 2004, pp. 1–8 (2004)
99. Jin, Y., Sendhoff, B., Koerner, E.: Evolutionary multi-objective optimization for simultaneous generation of signal-type and symbol-type representations. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 752–766. Springer, Heidelberg (2005)
100. Jin, Y., von Seelen, W., Sendhoff, B.: On generating FC3 fuzzy rule systems from data using evolution strategies. IEEE Trans. on Syst. Man and Cybern. - Part B 29, 829–845 (1999)
101. Karr, C.L.: Design of an adaptive fuzzy logic controller using a genetic algorithm. In: Proc. of 4th ICGA, pp. 450–457 (1991)
102. Karr, C.L., Gentry, E.J.: Fuzzy control of pH using genetic algorithms. IEEE Trans. on Fuzzy Syst. 1, 46–53 (1993)
103. Kaya, M.: Multi-objective genetic algorithm based approaches for mining optimized fuzzy association rules. Soft Comput. 10, 578–586 (2006)
104. Khara, V., Yao, X., Deb, K.: Performance scaling of multi-objective evolutionary algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003, vol. 2632, pp. 376–390. Springer, Heidelberg (2003)
105. Kim, H.S., Roschke, P.N.: Fuzzy control of base-isolation system using multi-objective genetic algorithm. Comput-Aided Civil and Infrast. Eng. 21, 436–449 (2006)
106. Kosko, B.: Fuzzy systems as universal approximators. In: Proc of FUZZ-IEEE 1992, pp. 1153–1162 (1992)
107. Kupinski, M.A., Anastasio, M.A.: Multiobjective genetic optimization of diagnostic classifiers with implications for generating receiver operating characteristic curve. IEEE Trans. on Med. Imaging 18, 675–685 (1999)
108. Li, H., Zhang, Q.F.: A multiobjective differential evolution based on decomposition for multiobjective optimization with variable linkages. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 583–592. Springer, Heidelberg (2006)
109. Liu, H., Motoda, H.: Feature selection for knowledge discovery and data mining. Kluwer Academic Publishers, Norwell (1998)
110. Llora, X., Goldberg, D.E.: Bounding the effect of noise in multiobjective learning classifier systems. Evol. Comput. 11, 278–297 (2003)
111. Mendel, J.M.: Fuzzy-logic systems for engineering - A tutorial. In: Proceedings of IEEE, vol. 83, pp. 345–377 (1995)
112. Miettinen, K.: Nonlinear multiobjective optimization. Kluwer, Boston (1998)
113. Mikut, R., Jakel, J., Groll, L.: Interpretability issues in data-based learning of fuzzy systems. Fuzzy Sets and Syst. 150, 179–197 (2005)
114. Muhlenbein, H., Schomisch, M., Born, J.: The parallel genetic algorithm as function optimizer. Parallel. Comput. 17, 619–632 (1991)
115. Nauck, D., Klawonn, F., Kruse, R.: Foundations of neuro-fuzzy systems. John Wiley & Sons, New York (1997)
116. Nauck, D., Kruse, R.: Obtaining interpretable fuzzy classification rules from medical data. Artif. Intell. in Med. 16, 149–169 (1999)
117. Nojima, Y., Ishibuchi, H.: Genetic rule selection with a multi-classifier coding scheme for ensemble classifier design. Int. J. of Hybrid Intell. Syst. 4, 157–169 (2007)
118. Nojima, Y., Ishibuchi, H., Kuwajima, I.: Parallel distributed genetic fuzzy rule selection. Soft Comput. (2008) (in press)
119. Oliveira, L.S., Morita, M., Sabourin, R.: Multi-objective genetic algorithms to create ensemble of classifiers. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 592–606. Springer, Heidelberg (2005)

120. Oliveira, L.S., Sabourin, R., Bortolozzi, F., Suen, C.Y.: Feature selection using multi-objective genetic algorithms for handwritten digit recognition. In: Proc. of ICPR 2002, pp. 568–571 (2002)
121. Oliveira, L.S., Sabourin, R., Bortolozzi, F., Suen, C.Y.: A methodology for feature selection using multi-objective genetic algorithms for handwritten digit string recognition. Int. J. of Pattern Recognit. and Artif. Intell. 17, 903–930 (2003)
122. Oliveira, L.S., Sabourin, R., Bortolozzi, F., Suen, C.Y.: Feature selection for ensembles: A hierarchical multi-objective genetic algorithm approach. In: Proc. of ICDAR 2003, pp. 676–680 (2003)
123. Parodi, A., Bonelli, P.: A new approach to fuzzy classifier systems. In: Proc. of 5th ICGA, pp. 223–230 (1993)
124. Pulkkinen, P., Koivisto, H.: Fuzzy classifier identification using decision tree and multiobjective evolutionary algorithms. Int. J. of Approx. Reason 48, 526–543 (2008)
125. Purshouse, R.C., Fleming, P.J.: Evolutionary many-objective optimization: An exploratory analysis. In: Proc. of CEC 2003, pp. 2066–2073 (2003)
126. Reynolds, A., de la Iglesia, B.: Rule induction using multi-objective metaheuristics: Encouraging rule diversity. In: Proc of IJCNN 2006, pp. 6375–6382 (2006)
127. Reynolds, A.P., de la Iglesia, B.: Rule induction for classification using multi-objective genetic programming. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 516–530. Springer, Heidelberg (2007)
128. Robic, T., Filipic, B.: DEMO: Differential evolution for multiobjective optimization. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 520–533. Springer, Heidelberg (2005)
129. Rodriguez-Vazquez, K., Fonseca, C.M., Fleming, P.J.: Multiobjective genetic programming: A nonlinear system identification application. In: Proc. of GP-97LB, pp. 207–212 (1997)
130. Roubos, H., Setnes, M.: Compact and transparent fuzzy models and classifiers through iterative complexity reduction. IEEE Trans. on Fuzzy Syst. 9, 516–524 (2001)
131. Rumelhart, D.E., McClelland, J.L., PDP Research Group: Parellel distributed processing. MIT Press, Cambridge (1986)
132. Setnes, M., Babuska, R., Verbruggen, B.: Rule-based modeling: Precision and transparency. IEEE Trans. on Syst. Man and Cybern. - Part C 28, 165–169 (1998)
133. Setnes, M., Roubos, H.: GA-based modeling and classification: Complexity and performance. IEEE Trans. on Fuzzy Syst. 8, 509–522 (2000)
134. Setzkorn, C., Paton, R.C.: On the use of multi-objective evolutionary algorithms for the induction of fuzzy classification rule systems. BioSyst. 81, 101–112 (2005)
135. Stewart, P., Stone, D.A., Fleming, P.J.: Design of robust fuzzy-logic control systems by multi-objective evolutionary methods with hardware in the loop. Eng. Appl. of Artif. Intell. 17, 275–284 (2004)
136. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. on Syst. Man and Cybern. 15, 116–132 (1985)
137. Thrift, P.: Fuzzy logic synthesis with genetic algorithms. In: Proc. of 4th ICGA, pp. 509–513 (1991)
138. Tsang, C.H., Kwong, S., Wang, H.L.: Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. Pattern Recognit. 40, 2373–2391 (2007)
139. Valenzuela-Rendon, M.: The fuzzy classifier system: A classifier system for continuously varying variables. In: Proc. of 4th ICGA, pp. 346–353 (1991)
140. Wang, H., Kwong, S., Jin, Y., Wei, W., Man, K.F.: Agent-based evolutionary approach for interpretable rule-based knowledge extraction. IEEE Trans. on Syst. Man and Cybern. - Part C 35, 143–155 (2005)

141. Wang, H., Kwong, S., Jin, Y., Wei, W., Man, K.F.: Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction. Fuzzy Sets and Syst. 149, 149–186 (2005)
142. Wang, L.X., Mendel, J.M.: Generating fuzzy rules by learning from examples. IEEE Trans. on Syst. Man and Cybern. 22, 1414–1427 (1992)
143. Wang, L.X., Mendel, J.M.: Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. IEEE Trans. on Neural Netw. 3, 807–814 (1992)
144. Xing, Z.Y., Zhang, Y., Hou, Y.L., Jia, L.M.: On generating fuzzy systems based on Pareto multi-objective cooperative coevolutionary algorithm. Int. J. of Control Autom. and Syst. 5, 444–455 (2007)
145. Yen, J., Wang, L., Gillespie, G.W.: Improving the interpretability of TSK fuzzy models by combining global learning and local learning. IEEE Trans. on Fuzzy Syst. 6, 530–537 (1998)
146. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithms. TIK-Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich (2001)
147. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Trans. on Evol. Comput. 3, 257–271 (1999)
148. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. on Evol. Comput. 7, 117–132 (2003)

Part III
Adaptive Solution Schemes

Exploring Hyper-heuristic Methodologies with Genetic Programming

Edmund K. Burke, Mathew R. Hyde, Graham Kendall, Gabriela Ochoa,
Ender Ozcan, and John R. Woodward*

Abstract. Hyper-heuristics represent a novel search methodology that is motivated by the goal of automating the process of selecting or combining simpler heuristics in order to solve hard computational search problems. An extension of the original hyper-heuristic idea is to generate new heuristics which are not currently known. These approaches operate on a search space of heuristics rather than directly on a search space of solutions to the underlying problem which is the case with most meta-heuristics implementations. In the majority of hyper-heuristic studies so far, a framework is provided with a set of human designed heuristics, taken from the literature, and with good measures of performance in practice. A less well studied approach aims to *generate* new heuristics from a set of potential heuristic *components*. The purpose of this chapter is to discuss this class of hyper-heuristics, in which Genetic Programming is the most widely used methodology. A detailed discussion is presented including the steps needed to apply this technique, some representative case studies, a literature review of related work, and a discussion of relevant issues. Our aim is to convey the exciting potential of this innovative approach for automating the heuristic design process.

1 Introduction

Heuristics for search problems can be thought of as "*rules of thumb*" for algorithmic problem solving [53]. They are not guaranteed to produce optimal solutions, rather,

Edmund K. Burke, Mathew R. Hyde, Graham Kendall, Gabriela Ochoa, and Ender Ozcan
University of Nottingham, School of Computer Science, Jubilee Campus, Wollaton Road
Nottingham, NG8 1BB, UK
e-mail: {ekb,mrh,gxk,gxo,exo}@cs.nott.ac.uk

John R. Woodward
University of Nottingham, 199 Taikang East Road, Ningbo 315100, China
e-mail: john.woodward@nottingham.edu.cn

* Corresponding author.

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 177–201.
springerlink.com

the goal is to quickly generate good quality solutions. They are often used when exact methods are unable to be employed in a feasible amount of time. Genetic Programming is a method of searching a space of computer programs, and therefore is an automatic way of producing programs. This chapter looks at the use of Genetic Programming to automatically generate heuristics for a given problem domain. A knowledge of Genetic Programming is assumed, and while a brief introduction is given, readers unfamiliar with the methodology are referred to suitable tutorials and textbooks.

1.1 The Need for Heuristics

Combinatorial problems arise in many disciplines such as artificial intelligence, logistics, operational research, finance and bioinformatics. Prominent examples are tasks such as finding shortest round trips in graphs (the travelling salesman problem), finding models of propositional formulae (Boolean satisfiability), or determining the 3D structure of proteins (the protein folding problem). Other well-known combinatorial problems are found in scheduling, planning, resource and space allocation, cutting and packing, software and hardware design, and genome sequencing. These problems are concerned with finding assignments, orderings or groupings of a discrete set of objects that satisfy certain constraints [30].

Most real-world combinatorial problems such as scheduling and planning, are difficult to solve. The main difficulty arises from the extremely large and/or heavily constrained search spaces, and the noisy/dynamic nature of many real-world scenarios. In practice, we often deal with them using *heuristic methods*, which have no guarantee of optimality and that often incorporate stochastic elements. Over the years, a large variety of heuristic methods have been proposed and are widely applied. Often, heuristics are the result of years of work by a number of experts. An interesting question is how can we automate the design of heuristics, and it is this question which represents the underlying motivation for this chapter. *Hyper-heuristics* [9, 49, 53] are search methodologies for choosing or generating (combining, adapting) heuristics (or components of heuristics), in order to solve a range of optimisation problems. We begin by looking at hyper-heuristics employed across a broad spectrum of applications in more detail.

1.2 Hyper-heuristics

The main feature of hyper-heuristics is that they search a space of heuristics rather than a space of solutions directly. In this sense, they differ from most applications of meta-heuristics, although, of course, meta-heuristics can be (and have been) used as hyper-heuristics. The motivation behind hyper-heuristics is to raise the level of generality at which search methodologies operate. Introductions to hyper-heuristics can be found in [9, 53].

An important (very well known) observation which guides much hyper-heuristic research is that different heuristics have different strengths and weaknesses. A key idea is to use members of a set of known and reasonably understood heuristics to either: (i) transform the state of a problem (in a constructive strategy), or (ii) perform an improvement step (in a perturbative strategy). Such hyper-heuristics have been successfully applied to bin-packing [54], personnel scheduling [13, 17], timetabling [1, 13, 14, 15, 59], production scheduling [61], vehicle routing problems [52], and cutting stock [58]. Most of the hyper-heuristic approaches incorporate a learning mechanism to assist the selection of low-level heuristics during the solution process. Several learning strategies have been studied such as reinforcement learning [17, 46], Bayesian learning [45], learning classifier systems [54], and case based reasoning [15]. Several meta-heuristics have been applied as search methodologies upon the heuristic search space. Examples are tabu search [13, 14], genetic algorithms [23, 29, 58, 59, 61], and simulated annealing [4, 18, 52]. This chapter focusses on Genetic Programming as a hyper-heuristic for generating heuristics, given a set of heuristic components.

1.3 Genetic Programming

Computers do not program themselves; they need a qualified and experienced programmer who understands the complexity of the task. An alternative to paying a human programmer to design and debug a program, is to build a computer system to evolve a program. This may not only be cheaper, but has the advantage that progress can be made on problem domains where a human programmer may not even have a clear idea of what the programming task is, as there is no formal program specification. Instead, a partial description of the desired program's behaviour could be supplied in terms of its input-output behaviour.

Genetic Programming [41, 42], a branch of program synthesis, borrows ideas from the theory of natural evolution to produce programs. The main components of evolutionary computation are *inheritance* (crossover), *selection* and *variation* (mutation). Inheritance implies that the offspring have some resemblance to their parents as almost all of the offspring's genetic material comes from them. Selection means that some offspring are preferable to others, and it is this selection pressure which defines which individuals are fitter then others. Variation supplies fresh genetic material, so individuals containing genetic material which was not present in either of the parents (or the wider gene pool) can be created. Evolutionary computation can be thought of the interaction of these three components.

Informally, a population of computer programs is generated, and the genetically inspired operations of mutation and crossover are applied repeatedly in order to produce new computer programs. These programs are tested against a *fitness function*, that determines which ones are more likely to survive to future generations. The fittest programs are more likely to be selected to continue in the evolutionary process (i.e. survival of the fittest).

More formally, a multiset of computer programs is generated. Programs are transformed by a number of operations, which typically take one or two computer programs as inputs. A fitness function assigns a value to each program (typically depending on its performance on the problem). A selection function generates a new multiset of programs from the previous multiset. This process is repeated until a termination condition is satisfied. In other words, Genetic Programming is a method of generating syntactically valid programs, according to some predefined grammar, and a fitness function is used to decide which programs are better suited to the task at hand.

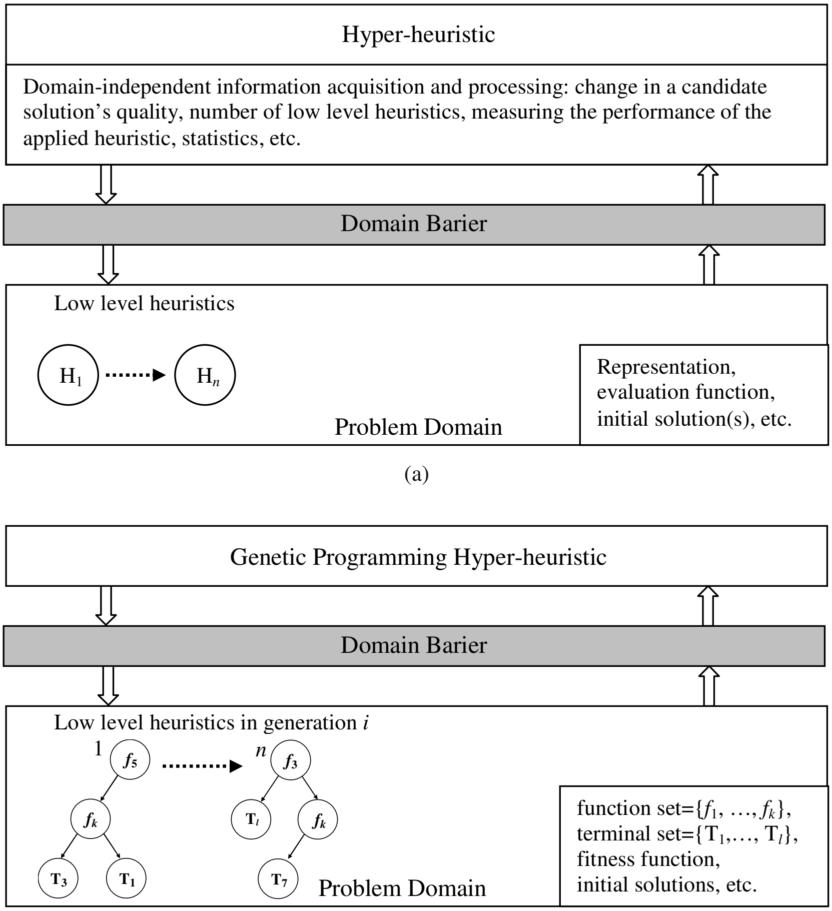
In Genetic Programming, the programs that comprise the population are traditionally represented as tree structures. There are other program structures which can be evolved, such as linear sequences of instructions or grammars. We will briefly introduce tree-based Genetic Programming. Each node in the tree returns a value to its parent node. The leaf nodes are usually input variables providing information about the problem state, or numeric constants. The internal nodes have one or more children nodes, and they return a value obtained by performing operations on the values returned by their children nodes. The trees' internal nodes are referred to as *functions*, and leaf nodes are referred to as *terminals*.

A number of decisions needs to be made before a Genetic Programming run is started. This includes the setting of parameters, such as the size of the population, and the termination condition (which is typically the number of generations). It also includes such as the *function set* and *terminal set*, along with the fitness function, which ultimately drives the evolutionary process. The terminal set is the set of nodes that can be the leaf nodes of the program tree, and as such, they take no arguments. They are the mechanism through which the problem state is made available to the program, and they act as input variables, changing their value as the problem state changes. The example of evolving a program to control a robot to clean a floor is given in [42]. The terminals may be defined as the movements that the robot can make, such as 'turn right', 'turn left', and 'move forward'. Other terminals may provide sensory information, such as how far an obstacle is from the robot. On the other hand, the function set is the set of operations that can be represented by the internal nodes of the tree. For example, they can be arithmetic operators, Boolean logic operators, or conditional operators. The functions of a Genetic Programming tree manipulate the values supplied to the program by the terminals, and as such their defining feature is that they take one or more arguments, which can be the values returned by terminal nodes, or other function nodes.

There is an important distinction which can be drawn between an optimisation problem and a learning problem. In the former, we seek the highest quality solution with respect to some evaluation function. An example is the minimisation of a function, where we seek a value of x such that $f(x)$ is a minimum. In the latter, we seek a solution which optimises the value of a target function on the validation data, which is independent of the training data. For example, consider function regression, where we seek a representation of $f(x)$, given a set of training data, but tested on a second set of data to confirm its ability to generalise. Typically, Genetic Programming is used as described in the second example. However, as we shall see, this distinction is apparent when we consider the difference between *reusable* heuristics (which need to be tested on a second set of examples to confirm their status as reusable heuristics), and *disposable* heuristics (which are only used on a single set of examples, without reuse in mind). For more details, see [42].

There are numerous tutorials, introductory articles and text books on Genetic Programming. See the series of books by Koza [38, 39, 40, 41] and the book by Banzhaf et al. [5]. Also [42] and [50] are more recent introductory texts. Introductory articles can also be found in most current textbooks on machine learning.

Genetic Programming can be employed as a hyper-heuristic. It can operate on a set of terminals and functions at the meta-level. Figure 1(a) shows a standard hyper-heuristic framework presented in [9, 17]. Figure 1(b) shows how Genetic Programming might be employed in this capacity. The base-level of a Genetic Programming hyper-heuristic includes the concrete functions and terminals associated with the problem. Across the domain barrier, abstract functions and terminals in the meta-level can be mapped to concrete functions and terminals in the base-level.


Hyper-heuristic
Domain-independent information acquisition and processing: change in a candidate solution’s quality, number of low level heuristics, measuring the performance of the applied heuristic, statistics, etc.
Domain Barier
Low level heuristics
H_1
H_n
Problem Domain
Representation, evaluation function, initial solution(s), etc.
Genetic Programming Hyper-heuristic
Domain Barier
Low level heuristics in generation *i*
1
f_5
f_k
T_3
T_1
n
f_3
T_l
f_k
T_7
Problem Domain
function set=$\{f_1, \ldots, f_k\}$,
terminal set=$\{T_1, \ldots, T_l\}$,
fitness function,
initial solutions, etc.

(a)

Fig. 1 (a) A generic and (b) a Genetic Programming hyper-heuristic framework

1.4 Chapter Outline

The outline of the remainder of this chapter is as follows. In Section 2, the use of Genetic Programming as a Hyper-heuristic is introduced. In Section 3, two cases studies are examined, namely the applications of Boolean Satisfiability and Online Bin Packing. In Section 4, some of the current literature concerning the automatic generation of heuristics is covered. Section 5 summarises and concludes the chapter.

2 Genetic Programming as a Hyper-heuristic

In this section, we examine a number of issues concerning the use and suitability of Genetic Programming to generate heuristics. A fundamental point concerning the scalability of this method is stated. As this methodology borrows ideas from human designed heuristics, which are then used as primitives to construct the search space of Genetic Programming, we are then in the enviable position of being able to guarantee heuristics which perform at least as good as human designed heuristics. Finally, we outline the basic approach to using Genetic Programming to generate heuristics.

2.1 Suitability of Genetic Programming as a Hyper-heuristic

A number of authors [5, 38, 39, 40, 41, 42] have pointed out the suitability of Genetic Programming over other machine learning methods to automatically produce heuristics. We list these advantages here (in no particular order).

- Genetic Programming has a variable length encoding. Often, we do not know (in advance) the optimal length of representation for heuristics for the given problem domain.
- Genetic Programming produces executable data structures. Heuristics are typically expressed as programs or algorithms.
- Humans can easily identify the good features of the problem domain which form the terminal set of a Genetic Programming approach.
- Human designed heuristics can readily be expressed in the language used to construct the search space of Genetic Programming. A function set, relevant to the problem domain can be determined without too much difficulty. In addition, the Genetic Programming system could also be supplemented with a grammar.

Of course, there are a number of disadvantages of using Genetic Programming to generate heuristics. For example, each time a Genetic Program is run it will give a different “best-of-run” heuristic, so it needs to be run multiple times, in order to gain a feel for the quality of the heuristics which it can produce. Other disadvantages include the, often unintuitive values for parameters, which are typically found through a trial and error process.

2.2 *The Basic Approach*

Given a problem domain, the application of Genetic Programming to generate heuristics can be undertaken as follows. Many of the steps described here are the same as those one would be required to go through in the construction of a normal Genetic Programming application (e.g. function regression). The main difference, which may not usually be required in a normal application of Genetic Programming, is to decide how the heuristic function is applied to the given problem domain.

1. **Examine currently used heuristics.** Here, we see if currently used heuristics can be described in a common framework, in which each existing heuristic is a special case. These could be either human designed or produced by other machine learning approaches. This step is not trivial and can involve the detailed understanding of the workings of a number of diverse existing heuristics, which may work in very different ways, in order to essentially arrive at the "big picture", or a generalisation of the heuristics used for the problem. Often, these human designed heuristics are the result of years of work by experts, so this process can be difficult.
2. **A framework for the heuristics to operate in.** We are concerned here with the question of how the heuristics are to be applied to an instance of the problem from the given domain. In general, this will be very different depending on the problem domain. It may be the case that many heuristics are applied in the same way, so it may be efficient to apply evolved heuristics in the same fashion. For example, many local search heuristics for the Boolean satisfiability problem fit into the same framework (see [25]).
3. **Decide on the terminal set.** Here, we decide on a set of variables which will express the state of the problem. These will appear as some of the terminals to the Genetic Programming system. Other terminals will also be needed. In particular, random constants are useful.
4. **Decide on the function set.** We need to know how the variables will be connected or composed together. This set of functions will form the function set of the Genetic Programming system. As with the problem of parameter setting (described below), it is worth revisiting this choice as the development progresses.
5. **Identify a fitness function.** A fitness function needs to be defined for the problem. Often, a simple naive fitness function does not perform very well, and introducing some parameters may help find a more suitable one.
6. **Run the Genetic Programming approach.** Often, a Genetic Programming system will not produce good solutions on a first run as poor parameters are chosen. This is especially the case with the novice practitioner. It is therefore essential that different parameter settings are thoroughly investigated.

3 Case Studies

We examine two examples in detail in order to illustrate the basic methodology (generating heuristics for Boolean satisfiability and online bin packing). In both cases, we describe the problem, a number of currently used human created heuristics, and some design questions about using Genetic Programming to generate heuristics. In the first example, evolving a local search heuristic for the Boolean satisfiability problem, a number of the design decisions (e.g. what variables are needed to express the problem, and a framework in which to express possible heuristics) seem reasonably straightforward, as similar choices were made by two independent authors [3, 25]. In the second example, these choices appear to be a little more difficult. The aim of this section, therefore, is to take the reader step by step through the process and raise a number of issues that will arise during the steps needed to apply Genetic Programming to generate heuristics. These domains have been chosen as they are well known problems, which both have published results concerning the automatic generation of heuristics.

3.1 Boolean Satisfiability – SAT

The Boolean satisfiability problem is the problem of finding the true/false assignments of a set of Boolean variables, to decide if a given propositional formula or expression (in conjunctive normal form) can be satisfied (i.e. does there exist values for the variables which make the expression evaluate to true). The problem is denoted as SAT. It is a classic NP-complete problem [27]. For example, the formula with three clauses $(a \vee b \vee \neg c) \wedge (\neg a \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$ is satisfiable as the formula evaluates to true when (a = true, b = false, c = true). However, the formula $a \wedge \neg b \wedge (\neg a \vee b) \wedge (a \vee b \vee c)$ is not satisfiable, as an assignment of the variables, such that the formula is true does not exist. A clause is referred to as *broken*, if all the variables in the clause are evaluated to be false under a given assignment. For example, in the formula $(\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg b) \wedge (\neg b \vee \neg c) \wedge (\neg a \vee \neg c)$, there are two broken clauses under the assignment (a = true , b = false, c = true): $(\neg a \vee b \vee \neg c)$ and $(\neg a \vee \neg c)$.

3.1.1 Existing Heuristics

Fukunaga [24] lists a number of well known local search heuristics which have been proposed in the SAT literature.

- *GSAT* selects a variable from the formula with the highest net gain. Ties are decided randomly.
- *HSAT* is the same as GSAT, but it decides ties in favour of maximum age, where age of a variable indicates the total number of bit-flips from the time when a variable was last inverted.
- *GWSAT(p)* (also known as "GSAT with random walk") randomly selects a variable with probability *p* in a randomly selected broken clause; otherwise, it is the same as GSAT.
- *Walksat(p)* picks a broken clause, and if any variable in the clause has a negative gain of 0, then it selects one of these to be flipped. Otherwise, it selects a random variable with probability *p* in the clause to flip, and selects a variable with

probability $(1-p)$ in the clause with minimal negative gain (breaking ties randomly). Otherwise, it selects a random variable with probability p in the clause to flip, and selects a variable with probability $(1-p)$ in the clause with minimal negative gain (breaking ties randomly).

Other heuristics, such as, *Novelty*, *Novelty*+ and *R-Novelty* are also discussed in [24].

3.1.2 Framework for Heuristics

Fukunaga [24] first examines the original local search heuristic GSAT, and also its many variants (including GSAT with random walk, and Walksat). Then, a template is identified which succinctly describes the most widely used SAT local search algorithms. This framework is also adopted by Bader-El-Den and Poli [3]. In this template, the set of Boolean variables are initially given random truth assignments. Repeatedly, a variable is chosen according to a variable selection heuristic and its value is inverted. This new assignment of values is then tested to see if it satisfies the Boolean expression. This is repeated until some cut off counter is reached. Notice that in this framework, only a single Boolean variable is selected to be inverted. An interesting alternative would be for the variable selection heuristic to return, not a single variable, but a subset of variables.

3.1.3 Identifying the Terminal Set

Fukunaga describes a number of factors in identifying which Boolean variables might be advantageous to invert. Let B_0 be the number of broken clauses in the expression, under the current variable assignment. Let B_1 be the number of broken clauses in the expression, under the current variable assignment, but when variable V is flipped. Let T, be the variable assignment and T^1 be the variable assignment when V is flipped. By looking at the number of clauses that become satisfied or unsatisfied when V is flipped, we can define a number of gain metrics. The net gain of V is $B_1 - B_0$. The negative gain of V is the number of clauses satisfied in T but broken in T^1. The positive gain of V is the number of clauses satisfied in T^1 but broken in T. Another example of a factor which can be used is the "age" of a variable (i.e. the number of inversions from the time when a variable was last inverted). These will form some of the terminals of the Genetic Programming system. For a complete list of terminals see [25].

3.1.4 Identifying the Function Set

Some heuristics are hybrid, in the sense that they are a combination of two existing heuristics. The "composition" (or blending) of two heuristics is achieved by first testing to determine if a condition is true, then if the test is passed apply heuristic1 else apply heuristic2. This composition operator therefore gives us a way to combine already existing heuristics. An example of the testing condition may simply

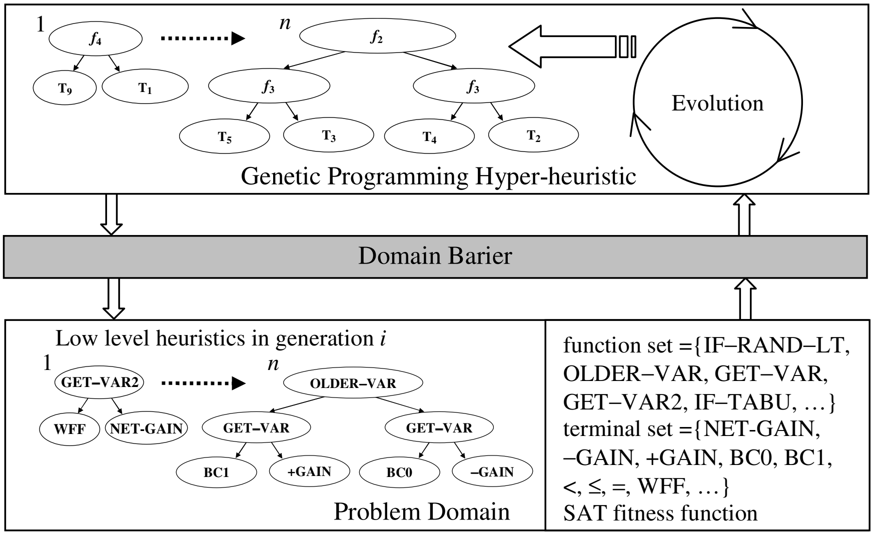

1
f_4
T_9
T_1
n
f_2
f_3
f_3
T_5
T_3
T_4
T_2
Evolution
Genetic Programming Hyper-heuristic
Domain Barier
Low level heuristics in generation i
1
n
GET–VAR2
WFF
NET-GAIN
OLDER–VAR
GET–VAR
GET–VAR
BC1
+GAIN
BC0
–GAIN
Problem Domain
function set ={IF–RAND–LT, OLDER–VAR, GET–VAR, GET–VAR2, IF–TABU, ...}
terminal set ={NET-GAIN, –GAIN, +GAIN, BC0, BC1, <, ≤, =, WFF, ...}
SAT fitness function

Fig. 2 Genetic Programming as a hyper-heuristic. At the meta-level Genetic Programming refers to abstract functions $\{f_1, f_2, ...\}$, and terminals $\{T_1, T_2, ...\}$. At the base-level these are given concrete meaning. For example, $f_1 =$ IF–RAND–LT , $f_2 =$ OLDER–VAR , $T_1 =$ NET–GAIN, $T_2 =$ –GAIN, etc

be "(random number $\leq$ 0.2)". Having identified a template for local search and a method of identifying the utility of inverting a given variable, Fukunaga then defined a language in which most of the previously human designed heuristics can be described, but more importantly, it can also be used to describe new novel heuristics. For a complete list of functions see [25].

3.1.5 Identifying a Fitness Function

The fitness function works as follows. First, the heuristic is tested on 200 problem instances consisting of 50 variables and 215 clauses. The heuristic is allowed 5000 inversions of the Boolean variables. If more than 130 of these local searches were successful, then the heuristic is run on 400 problem instances consisting of 100 variables with 430 clauses. The heuristic is allowed 20000 inversions of the Boolean variables. The idea of using smaller and larger problems, is that poor candidate heuristics can be culled early on (very much like brood selection, reported in [5]).

$$\begin{aligned} fitness = \; & \text{(number of 50 variable successes)} \\ & + 5\text{(number of 100 variable successes)} \\ & + 1/\text{(mean number of flips in successful runs)} \end{aligned} \tag{1}$$

The second term carries a weight of 5, as performance on these instances is much more important. In the case of a tie-break, the last term differentiates these heuristics. It should be noted that the fitness function takes a large number of parameters, and reasonable values for these should be arrived at with a little experimentation.

3.2 Online Bin Packing

The online bin packing problem can be described as the problem of being given a sequence of items and assigning each one to a bin as it arrives, such that the minimum number of bins is required [55]. There is an unlimited supply of bins, each with the same finite capacity which must not be exceeded. We do not know in advance either the sizes of the items, or the total number of items. This is in contrast to the offline version of the problem where the set of items to be packed is available from the start.

3.2.1 Existing Heuristics

A number of examples of heuristics used in the online bin packing problem are described below: In each case, if the item under consideration does not fit into an existing bin, then the item is placed in a new bin.

- *Best-Fit* [51]. Puts the item in the fullest bin which can accommodate it.
- *Worst-Fit* [16]. Puts the item in the emptiest bin which can accommodate it.
- *Almost-Worst-Fit* [16]. Puts the item in the second emptiest bin.
- *Next-Fit* [36]. Puts the item in the last available bin.
- *First-Fit* [36]. Puts the item in the left-most bin.

It should, of course, be noted that this list of heuristics is not exhaustive. The selection is simply intended to illustrate some of the currently available heuristics, and provide a background against which we can build a framework. The reader is referred to the following article if they are particularly interested in the domain of online bin packing . Here the HARMONIC algorithms are discussed (which include HARMONIC, REFINED HARMONIC, MODIFIED HARMONIC, MODIFIED HARMONIC 2, and HARMONIC+1). All of these algorithms are shown to be instances of a general class of algorithm, which they call SUPER HARMONIC.

3.2.2 A Framework for Heuristics

In [10, 11, 12], heuristics are evolved for the online bin packing problem. In the first paper [10], a number of existing heuristics are listed. Interestingly these heuristics do not fit neatly into a single framework. In this paper, the decision was made to apply the evolved heuristic to the bins and place the current item under consideration, into the first bin which receives a positive score. Using this method of applying heuristics to problem instances, a heuristic equivalent to the "first-fit" heuristic was evolved. The "first-fit" heuristic places an item in the first bin into which it fits (the order of the bins being the order in which they were opened). In this framework, the

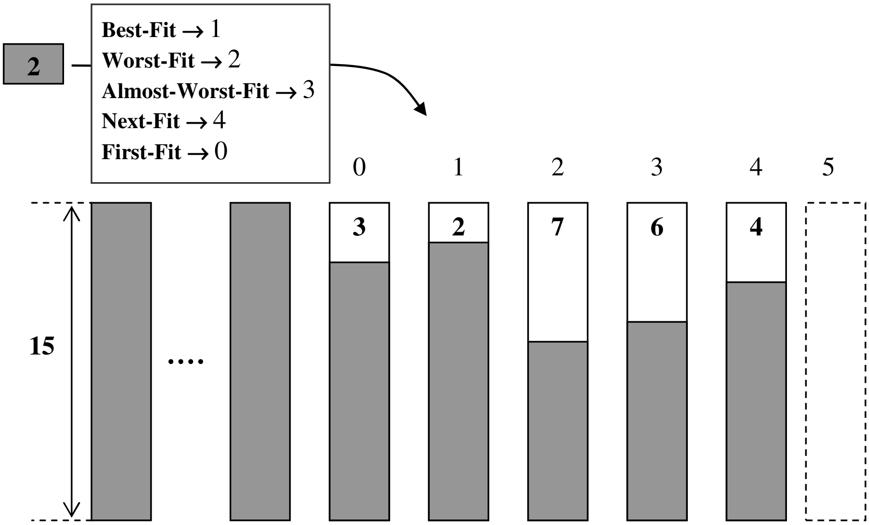


Fig. 3 The figure shows the chosen bin for a number of heuristics. The bin capacity is 15, and the space remaining in the open bins (in order of index 0, 1, 2, 3, 4, 5) is 3, 2, 7, 6, 4, 15. The current item to be packed has size 2. "Best-fit", for example would place the item in bin 1, leaving no space remaining. "First-fit", for example would place the item in bin 0, leaving 1 unit of space

```
For each item
  int binIndex := 0
  For each bin b in A
    output := evaluate Heuristic
    If (output > 0 )
      return binIndex
    End If
  End For
  place item in bin binIndex
End For
```

Fig. 4 An item is considered for each bin in turn, until a positive score is obtained. Thus the heuristic may not be evaluated on all bins, for a given item. The item is placed in the bin which gives the first positive score. This method of applying heuristics differs fundamentally from the method described in figure 5

heuristic may not be evaluated on all of the bins when an item is being placed (i.e. only the bins up until the bin that receives a positive score will be considered).

In [11], it was decided that the heuristic would be evaluated on all the open bins, and the item placed in the bin that receives the maximum score. This has the advantage that the heuristic is allowed to examine all of the bins (and, therefore, has more information available to it to make a potentially better decision). It also has the

```
For each item
  int currentMaximumScore = -∞
  int binIndex := 0
  For each bin b
    output := evaluate Heuristic
    If (output > currentMaximumScore )
      currentMaximumScore = output
      binIndex := b
    End If
  End For
  place item p in bin binIndex
return binIndex
```

Fig. 5 In this framework, the item is placed in the bin which gives the maximum score according to the heuristic. This method of applying heuristics differs fundamentally from the method described in figure 4

disadvantage that it will take longer on average to apply, as it will, in general, examine more bins (though this aspect of the evolved heuristic's performance was not studied). In this framework, heuristics were evolved which outperformed the human designed heuristic "best-fit". The "best-fit" heuristic places an item in the bin which has the least space remaining when the item is placed in the bin (i.e. it fits best in that bin).

The two search spaces created by these frameworks are very different. In the first case, the "first-fit" heuristic can be expressed, but "best-fit" cannot. In the second case, the "best-fit" heuristics can be expressed but "first-fit" cannot. The first framework cannot express "best-fit", as not all of the bins may be examined. That is, the evaluation of the heuristic is terminated as soon as a positive score is obtained. The second framework cannot express "first-fit" as a bin which receives a larger score may exist after one which receives a positive score. That is, an earlier bin may receive a smaller positive score, but this is overridden when a larger score is obtained. Further effort could be put into constructing a more general framework in which both of these heuristics could be expressed.

So far, just two frameworks have been considered which could be used to apply heuristics to the online bin packing problem. There are many different ways a heuristic could be applied.

- They can differ in the order in which the bins are examined. For example, left to right, right to left, or some other order.
- They can differ in the order we start to examine the bins. For example, start at a random bin and cycle through the bins until each bin has been examined, or start at some distance from the last bin that received an item.
- They can differ in the score used to decide which bin is employed. For example, place the item in the bin which got the second highest score, or alternatively place the item in the bin which gets the maximum then the next item in the bin that gets the minimum score; in effect we are switching between two placement strategies.

There is also the question of where to place an item when there is a draw between two bins (e.g. the item could be placed in a fresh bin, or it could be placed in a bin using an existing human designed heuristic). The point is that there are plenty of opportunities to design different ways of applying heuristic selection functions to a problem domain. Therefore, instead of presenting Genetic Programming with a single framework, it is possible to widen this and allow a number (or combination) of different frameworks for Genetic Programming to explore. One interesting way to tackle this would be to cooperatively co-evolve a population of heuristics and the frameworks in which they are applied.

It is also worthwhile pointing out that a heuristic evolved under one framework is unlikely to perform well under another framework, so a heuristic really consists of two parts; the heuristic function and the framework describing how the heuristic is applied to a problem instance. In Genetic Programming, we are usually just interested in the function represented by a program, and the program does not need a context (e.g. in the case of evolving electrical circuits, the program is the solution). However, if we are evolving heuristics, we need to provide a context or method of applying the Genetic Programming-program. This additional stage introduces a considerable difference.

3.2.3 Identifying the Terminal Set

The question of which variables to use to describe the state of a problem instance is also important, as these will form some of the "terminals" used in Genetic Programming. In the first stages of this work, the authors used the following variables; S the size of the current item, C the capacity of a bin (this is a constant for the problem) and, F the fullness of a bin (i.e. what is the total cost of all of the items occupying that bin).

It was later decided that three variables could be replaced by two; S the size of the current item and, E $(= C - F)$ the emptiness of a bin (i.e. how much space is remaining in the bin, or how much more cost can be allocated to it before it exceeds its capacity). These two variables are not as expressive as the first set, but are expressive enough to produce human competitive heuristics. The argument is that it is not the capacity or fullness of a bin which is the deciding factor of whether to put an item in a bin, but the remaining capacity, or emptiness E, of a bin. In fact, the capacity of a bin was fixed for the entire set of experiments, so could be considered as a constant. In other words, the output of the heuristic based on this pair of variables, could be semantically interpreted as how suitable is the current bin, given its emptiness, and the size of the item we would like to place in the bin.

This pair of variables can be replaced by a single variable, R $(= E - S)$ the space remaining in a bin if the current item were placed in the bin. The output of a heuristic based solely on this single variable could not be interpreted as in the previous case, but rather as the following question: If the current item were placed in the current bin, what is the utility of the remaining space?

So far, we have only considered variables describing the current item and current bin. However, there are other variables which could be introduced. Other examples of variables which could be stored are

- the item number (i.e. keep a counter of how many items have been packed so far)
- the minimum and maximum item size seen so far (as more items are packed, these bounds will diverge)
- the average and standard deviation of item size seen so far (these could provide a valuable source of statistical information on which to base future decisions).

All of this information can be made available to the evolving heuristic.

3.2.4 Identifying the Function Set

In [10], the function set $\{+, -, x, \%, abs, \leq\}$ was used, where *abs* is the absolute operator and % is protected divide. There are a few points worth considering with this chosen function set. Firstly, $\leq$ returns -1 or +1, rather than 0 or 1, which is normally associated with this relational operator. This was to satisfy the property of closure, that the output of any function in the function set, can be used as the input of any function in the function set. Secondly, this function set is sufficient to express some the human designed heuristics described (namely “first-fit” and “best-fit”).

Protected divide (%) is often used in Genetic Programming, as if the denominator is zero, then the function is undefined (i.e. its value tends to infinity). Typically, protected divide returns 0 or 1. However, this choice does not reflect the idea that the quotient could be a very large number. Thus, in [11], a much larger value was returned.

In [12], $\leq$ was removed from the function set as it was effectively redundant. This is because, as the evolved heuristic function is enveloped in a loop which returns the index of the maximum scoring bin, any test for ‘less than’ can be done by the loop. The aim of this discussion is to outline the difficulty in choosing a function set for the given problem domain.

3.2.5 Identifying a Fitness Function

The fitness function to determine the quality of a solution is shown in Equation 2 [22], where n = number of bins used, F_i = fullness of bin i, and C = bin capacity

$$Fitness := \begin{cases} high\ penalty\ value, & \text{if illegal solution} \\ 1 - \left(\frac{\sum_{i=1}^{n}(F_i/C)^k}{n}\right), & \text{if legal solution} \end{cases} \tag{2}$$

It returns a value between 0 and 1, with 0 being the best result where all bins are filled completely, and 1 representing completely empty bins.

In the bin packing problem, there are many different solutions which use the same number of bins. If the fitness function were simply the number of bins used, then there would be a plateau in the search space that is easily reached, but difficult to escape from [22]. Using equation 2 as a fitness function helps the evolutionary process by

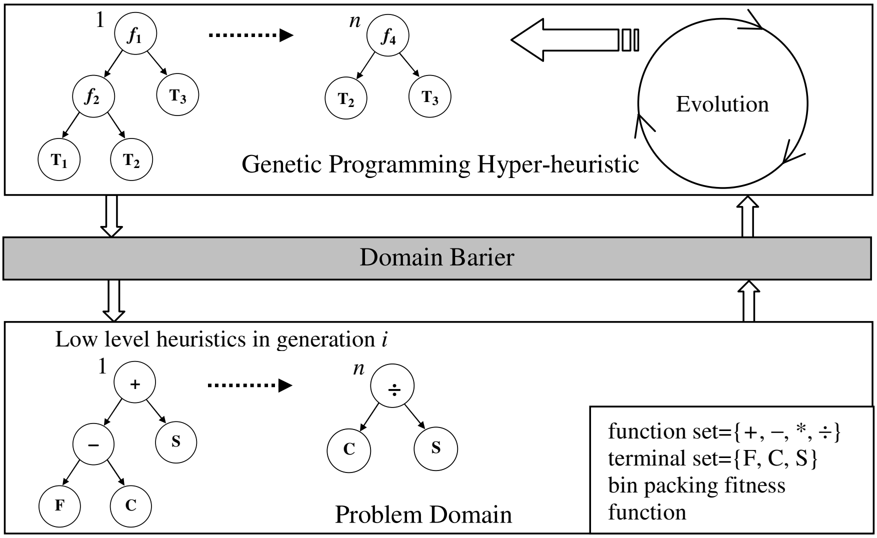


Fig. 6 Genetic Programming as a hyper-heuristic. At the meta-level Genetic Programming refers to abstract functions $\{f_1, f_2, ...\}$, and terminals $\{T_1, T_2, ...\}$. At the base-level these are given concrete meaning. For example, $f_1 = +$, $f_2 = -$, $T_1 =$ F, $T_2 =$ C, etc

differentiating between two solutions that use the same number of bins. The fitness function proposed by Falkenauer rewards solutions more if some bins are nearly full and others nearly empty, as opposed to all the bins being similarly filled.

The constant k in equation 2 determines how much of a premium is placed on nearly full bins. The higher the value of k, the more attention will be given to the almost filled bins at the expense of the more empty ones. A value of $k = 2$ was deemed to be the best in [22] so this is the value we use here.

4 Literature Review

In this section, we briefly discuss the area, in order to give the reader a flavour of what has been attempted to date. We include some work specifically using Genetic Programming as a hyper-heuristic. We also include some work on other areas which are similar in the sense that they use a meta-level in the learning system, and can tackle multiple problems. We now briefly review two areas of the machine learning literature which could also be considered in the context of hyper-heuristics. The first is learning to learn, and the second is self-adaptation.

4.1 Genetic Programming Hyper-heuristics for Generating Reusable Heuristics

Keller et al. [37] use a grammar to describe promising meta-heuristics for the travelling salesman problem. Primitives in the grammar may represent manually created meta-heuristics, low level heuristics, or component parts of them. There are a number of heuristics used in this system, including heuristics which swap two or three edges in the solution, and an iterative heuristic which executes another heuristic a maximum of 1000 times unless an improvement is seen. The execution of the meta-heuristic is a sequential execution of a list of heuristics and so generates a candidate solution to the given problem from a random initial route. Tours whose lengths are highly competitive with the best real-valued lengths from the literature are found using this grammar based Genetic Programming.

In a series of papers, Burke et al. [10, 11, 12] examine the viability of using Genetic Programming to evolve heuristics for the online bin packing problem. Given a sequence of items, each must be placed into a bin in the order it arrived. At each decision point, we are only given the size of the current item to be packed. In [10], an item is placed into the first bin which receives a positive score according to the evolved heuristic. Thus, the heuristic may not be evaluated for all bins, as it is terminated as soon as a positive score is obtained. This approach produces a heuristic which performs better than the human designed "first-fit" heuristic.

In [11], a similar approach is used. However, this time, the heuristic is allowed to examine all bins, and the item is placed in the bin which receives the maximum score. This produces a heuristic which is competitive with the human designed heuristic "best-fit". The difference between these two approaches, illustrates that the framework to evaluate the heuristics is a critical component of the overall system. In [11], the performance of heuristics on general and specialised problem classes is examined. They show that, as one problem class is more general than another, then the heuristic evolved on the more general class is more robust, but performs less well than the specialised heuristic on the specialised class of problem. This is, intuitively, what one would expect.

In [12], evolved heuristics are applied to much larger problem instances than they were trained on, but as the larger instances come from the same class as the smaller training instances, performance does not deteriorate and indeed, the approach consistently outperforms the human designed best-fit heuristic. The paper makes the important distinction between the nature of search spaces associated with direct and indirect methods. With direct methods, the size of the solution necessarily grows with the size of the problem instance, resulting in combinatorial explosion, for example, when the search space is a permutation. However, when the search space consists of programs or heuristics, the size of a program to solve a given class of problem is fixed as it is a generalisation of the solution to a class of problem (i.e. the solution to a class of problem is independent of the size of an instance).

Drechsler et al. [19], instead of directly evolving a solution, use Genetic Programming to develop a heuristic that is applied to the problem instance. Thus the typically large run-times associated with evolutionary runs have to be invested only once in the learning phase. The technique is applied to a problem of minimising Binary Decision Diagrams. They state that standard evolutionary techniques cannot be applied due to their large runtime. The best known algorithms used for variable ordering are exponential in time, thus heuristics are used. The heuristics which are developed by the designer often fail for specific classes of circuits. Thus it would be beneficial if the heuristics could learn from previous examples. An earlier paper is referred to where heuristics are learnt using a genetic algorithm [20], but it is pointed out that there are problems using a fixed length encoding to represent heuristics. Experiments show that high quality results are obtained that outperform previous methods, while keeping low run-times.

Fukunaga [24, 25] examines the problem domain of Boolean satisfiability (SAT). He shows that a number of well-known local search algorithms are combinations of variable selection primitives, and he introduces CLASS (Composite heuristic Learning Algorithm for SAT Search), an automated heuristic discovery system which generates variable selection heuristic functions. The learning task, therefore, is designing a variable selection heuristic as a meta-level optimisation problem.

Most of the standard SAT local search procedures can be described using the same template, which repeatedly chooses a variable to invert, and calculates the utility in doing so. Fukunaga identifies a number of common primitives used in human designed heuristics e.g. the gain of flipping a variable (i.e. the increase in the number of clauses in the formula) or the age of a variable (i.e. how long since it was last flipped). He states that *"it appears human researchers can readily identify interesting primitives that are relevant to variable selection, the task of combining these primitives into composite variable selection heuristics may benefit from automation"*. This, of course, is particularly relevant for Genetic Programming.

In the CLASS language, which was shown to able to express human designed heuristics, a composition operator is used which takes two heuristics and combines them using a conditional if statement. The intuition behind this operator is that the resulting heuristic blends the behaviour of the two component heuristics. The importance of this composition operator is that it maintains the convergence properties of the individual heuristics, which is not true if Genetic Programming operators were used. CLASS successfully generates a new variable selection heuristic, which is competitive with the best-known GSAT/Walksat-based algorithms. All three learnt heuristics were shown to scale and generalise well on larger random instances; generalisation to other problem classes varied.

Geiger et al. [28] present an innovative approach called SCRUPLES (scheduling rule discovery and parallel learning system) which is capable of automatically discovering effective dispatching rules. The claim is made that this is a significant step beyond current applications of artificial intelligence to production scheduling, which are mainly based on learning to select a given rule from among a number of candidates rather than identifying new and potentially more effective rules. The rules discovered are competitive with those in the literature. They state that a review of the literature shows no existing work where priority dispatching rules are discovered through search. They employ Genetic Programming, as each dispatching rule is viewed as a program. They point out that, Genetic Programming has a key advantage over more conventional techniques such as genetic algorithms and neural networks, which deal with fixed sized data structures. Whereas Genetic Programming can discover rules of varying length and for many problems of interest, such as scheduling problems, the complexity of an algorithm which will produce the correct solution is not known a-priori. The learning system has the ability to learn the best dispatching rule to solve the single unit capacity machine-scheduling problem. For the cases where no dispatching rules produced optimal solutions, the learning system discovers rules that perform no worse than the known rules.

Stephenson et al. [57] apply Genetic Programming to optimise priority or cost functions associated with two compiler heuristics; predicted hyper block formation (i.e. branch removal via prediction) and register allocation. Put simply, priority functions prioritise the options available to a compiler algorithm. Stephenson et al. [57] state "*Genetic Programming is eminently suited to optimising priority functions because they are best represented as executable expressions*". A caching strategy, is a priority function that determines which program memory locations to assign to cache, in order to minimise the number of times the main memory must be accessed. The human designed "least recently used" priority function is outperformed by results obtained by Genetic Programming. They make the point that by evolving compiler heuristics, and not the applications themselves, we need only apply our process once, which is in contrast to an approach using genetic algorithms. In addition they emphasise that compiler writers have to tediously fine tune priority functions to achieve suitable performance, whereas with this method, this is effectively automated.

4.2 Genetic Programming Hyper-heuristics for Generating Disposable Heuristics

Bader-El-Din et al. [3] present a Genetic Programming hyper-heuristic framework for the 3-SAT problem domain. Their aim is not to evolve reusable heuristics, but to solve a set of problem instances. The evolved heuristics are essentially disposed of and are considered as a by-product of the evolutionary process. Human designed heuristics are broken down into their constituent parts, and a grammar is used to capture the structure of how the constituents relate to each other. The constituent parts, along with the grammar, are used to construct a search space, which contains (by definition) the human designed heuristics. The resulting space is searched using Genetic Programming. Although the initial population of heuristics were randomly generated and included no handcrafted heuristics as primitives, individuals representing such heuristics were created in the initial population in almost all experiments (i.e. heuristics equivalent to human designed heuristics were found by random search). This is due to their simple representation in the grammar defined in the system.

4.3 Learning to Learn

Learning to learn [60] is similar to using Genetic Programming as a hyper-heuristic to solve a class of problem. Rather than trying to learn a single instance of a problem, a class of related problems is tackled. The key idea is to have two explicit levels in the learning algorithm, a meta-level and a base-level. The base-level is associated with learning a function, just like regular supervised learning in the single task case. The meta-level is responsible for learning properties of these functions (i.e. invariants or similarities between the problem instances). Thus, the meta-level is responsible for learning across the distribution of problems. Any machine-learning paradigm could be used at the base-level or meta-level.

4.4 The Teacher System

An interesting related project at the interface between machine learning and engineering was termed "Teacher" [62, 63] (an acronym for TEchniques for the Automated Creation of HEuRistics), which was designed as a system for learning and generalising heuristics used in problem solving. The objective was to find, under resource constraints, improved heuristic methods as compared to existing ones, in applications with little (or non-existent) domain knowledge. The Teacher system employed a genetic-based machine learning approach, and was successfully applied to several domains such as: process mapping, load balancing on a network of workstations, circuit placement, routing and testing. The system addressed five important general issues in learning heuristics [62]: "(1) *decomposition* of a problem solver into smaller components and integration of heuristic methods designed for each smaller component; (2) *classification* of an application domain into subdomains so that the performance can be evaluated statistically for each; (3) *generation* of new and improved heuristic methods based on past performance information and heuristics generated; (4) *evaluation* of each heuristic method's performance; and (5) *performance generalization* to find heuristic methods that perform well across the entire application domain".

4.5 Related Areas

Heuristic search represents a major research activity at the interface of Operational Research and Artificial Intelligence. It provides the core engine for real-world applications as diverse as timetabling, planning, personnel and production scheduling, cutting and packing, space allocation, and protein folding. Several researchers have recognised that a promising direction for developing improved and automated search techniques is to integrate learning components that can adaptively guide the search. Many techniques have independently arisen in recent years that exploit either some form of learning, or search on a search space of algorithm configuration, to improve problem-solving and decision making. A detailed review of these techniques is beyond the scope of this chapter. However, we mention here some related areas of research: adaptation and self-adaptation of algorithm parameters [2, 21, 32], algorithm configuration [33], racing algorithms [8], reactive search [6, 7], adaptive memetic algorithms [34, 35, 43, 44, 47, 48, 56], and algorithm portfolios [26, 31].

5 Summary and Conclusion

Often, in the field of computational search, a single problem (sometimes even a single instance) is tackled. This chapter describes work which is motivated by the goal of moving away from this situation. The work described here is attempting to "raise the level of generality". This offers a number of long term advantages. In particular, we can obtain more general systems rather than problem specific approaches. Moreover, we can achieve this more cheaply in terms of resource(s) used; i.e. a computer system is much cheaper to run than a team of heuristic designers is to employ.

5.1 The Need for Automatic Heuristic Generation

Real-world intractable problems demand the use of heuristics if progress is to be made in reasonable time. Therefore, the practical importance of heuristics is unquestionable, and how heuristics are produced then becomes an important scientific question. Many of the current heuristics in use today are the result of years of study by experts with specialist knowledge of the domain area. Therefore, one may pose the question;

> *Instead of getting experts to design heuristics, perhaps they would be better employed designing a search space of heuristics (i.e. all possible heuristics or a promising subset of heuristics) and a framework in which the heuristics operate, and letting a computer take over the task of searching for the best ones.*

This approach shows a clear division of labour; Humans, taking on the innovative and creative task of defining a search space. Computers take on the chore of searching this vast space. Due to the fact that humans often still need to play an important part in this process, we should strictly refer to this methodology as a *semi*-automated process.

One of the advantages of this methodology is that if the problem specification were to change, the experts who engage in hand designing heuristics, would probably have to return to the drawing board, possibly approaching the problem from scratch again. This would also be the situation with the search for automatically designed heuristics, with one important difference. As the search process is automated this would largely reduce the cost of having to create a new set of heuristics. In essence, by employing a method automated at the meta-level, the system could be designed to tune itself to the new problem class presented to it.

A paradigm shift has started to occur in search methodologies over the past few years. Instead of taking the rather short term approach of tackling single problems, there is a growing body of work which is adopting the more long term approach of tackling the general problem, and providing a more general solution.

5.2 Supplementing Human Designed Heuristics

Typically in the "meta-heuristics to choose heuristics" framework, the heuristics are human designed, and therefore have all the strengths of human designed heuristics, but also all of the weaknesses. In contrast, machine generated heuristics will have their own strengths and weaknesses. Thus, as one of the motives of hyper-heuristics is to combine heuristics, this would offer a method where manually and automatically designed heuristics can be used side by side. It may also be possible to evolve heuristics specifically to complement human designed heuristics in a hyper-heuristic context, where an individual heuristic does not need to be good on its own, but is a good team player in the environment of the other heuristics. Again this is another example of cooperation between humans and computers.

References

1. Asmuni, H., Burke, E.K., Garibaldi, J.M., Mccollum, B.: A novel fuzzy approach to evaluate the quality of examination timetabling. In: Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, pp. 82–102 (2006)
2. Bäck, T.: An overview of parameter control methods by self-adaption in evolutionary algorithms. Fundam. Inf. 35(1-4), 51–66 (1998)
3. Bader-El-Din, M.B., Poli, R.: Generating SAT local-search heuristics using a GP hyper-heuristic framework. In: Monmarché, N., Talbi, E.-G., Collet, P., Schoenauer, M., Lutton, E. (eds.) EA 2007. LNCS, vol. 4926, pp. 37–49. Springer, Heidelberg (2008)
4. Bai, R., Kendall, G.: An investigation of automated planograms using a simulated annealing based hyper-heuristic. In: Ibaraki, T., Nonobe, K., Yagiura, M. (eds.) Metaheuristics: Progress as Real Problem Solver. Operations Research/Computer Science Interface Serices, vol. 32, pp. 87–108. Springer, Heidelberg (2005)
5. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann, San Francisco (1998)
6. Battiti, R.: Reactive search: Toward self–tuning heuristics. In: Rayward-Smith, V.J., Osman, I.H., Reeves, C.R., Smith, G.D. (eds.) Modern Heuristic Search Methods, pp. 61–83. John Wiley & Sons Ltd, Chichester (1996)
7. Battiti, R., Brunato, M.: Reactive search: Machine learning for memory-based heuristics. In: Gonzalez, T.F. (ed.) Approximation Algorithms and Metaheuristics, ch. 21, pp. 1–17. Taylor and Francis Books/CRC Press, Washington (2007)
8. Birattari, M.: The problem of tuning metaheuristics as seen from a machine learning perspective. Ph.D. thesis, Universite Libre de Bruxelles (2004)
9. Burke, E.K., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S.: Hyper-heuristics: An emerging direction in modern search technology. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 457–474. Kluwer, Dordrecht (2003)
10. Burke, E.K., Hyde, M.R., Kendall, G.: Evolving bin packing heuristics with genetic programming. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 860–869. Springer, Heidelberg (2006)
11. Burke, E.K., Hyde, M.R., Kendall, G., Woodward, J.: Automatic heuristic generation with genetic programming: evolving a jack-of-all-trades or a master of one. In: Thierens, D., et al. (eds.) Proceedings of the 9th annual conference on Genetic and evolutionary computation GECCO 2007, vol. 2, pp. 1559–1565. ACM Press, London (2007)
12. Burke, E.K., Hyde, M.R., Kendall, G., Woodward, J.R.: The scalability of evolved on line bin packing heuristics. In: Srinivasan, D., Wang, L. (eds.) 2007 IEEE Congress on Evolutionary Computation, pp. 2530–2537. IEEE Computational Intelligence Society/IEEE Press, Singapore (2007)

13. Burke, E.K., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling and rostering. J. of Heuristics 9(6), 451–470 (2003)
14. Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R.: A graph-based hyper-heuristic for educational timetabling problems. Eur. J. of Oper. Res. 176, 177–192 (2007)
15. Burke, E.K., Petrovic, S., Qu, R.: Case based heuristic selection for timetabling problems. J. of Sched. 9(2), 115–132 (2006)
16. Coffman Jr., E.G., Galambos, G., Martello, S., Vigo, D.: Bin packing approximation algorithms: Combinatorial analysis. In: Du, D.Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization, pp. 151–207. Kluwer, Dordrecht (1998)
17. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001) (selected papers)
18. Dowsland, K.A., Soubeiga, E., Burke, E.K.: A simulated annealing hyper-heuristic for determining shipper sizes. Eur. J. of Oper. Res. 179(3), 759–774 (2007)
19. Drechsler, N., Schmiedle, F., Grosse, D., Drechsler, R.: Heuristic learning based on genetic programming. In: Miller, J., Tomassini, M., Lanzi, P.L., Ryan, C., Tetamanzi, A.G.B., Langdon, W.B. (eds.) EuroGP 2001. LNCS, vol. 2038, pp. 1–10. Springer, Heidelberg (2001)
20. Drechsler, R., Becker, B.: Learning heuristics by genetic algorithms. In: ASP-DAC 1995: Proceedings of the 1995 conference on Asia Pacific design automation (CD-ROM), p. 53. ACM, New York (1995)
21. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in Evolutionary Algorithms. IEEE Trans. on Evol. Comput. 3(2), 124–141 (1999)
22. Falkenauer, E., Delchambre, A.: A genetic algorithm for bin packing and line balancing. In: Proc. of the IEEE 1992 International Conference on Robotics and Automation, pp. 1186–1192 (1992)
23. Fang, H.L., Ross, P., Corne, D.: A promising hybrid GA/heuristic approach for open-shop scheduling problems. In: Eur. Conference on Artificial Intelligence (ECAI 2004), pp. 590–594 (1994)
24. Fukunaga, A.S.: Automated discovery of composite sat variable-selection heuristics. In: AAAI/IAAI, pp. 641–648 (2002)
25. Fukunaga, A.S.: Automated discovery of local search heuristics for satisfiability testing. Evol. Comput. 16(1), 31–61 (2008)
26. Gagliolo, M., Schmidhuber, J.: Learning dynamic algorithm portfolios. Ann. of Math. and Artif. Intell. 47(3-4), 295–328 (2006)
27. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Series of Books in the Mathematical Sciences. W. H. Freeman, New York (1979)
28. Geiger, C.D., Uzsoy, R., Aytug, H.: Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach. J. of Sched. 9(1), 7–34 (2006)
29. Hart, E., Ross, P., Nelson, J.: Solving a real-world problem using an evolving heuristically driven schedule builder. Evol. Comput. 6(1), 61–80 (1998)
30. Hoos, H.H., Sttzle, T.: Stochastic Local Search: Foundations and Applications. Morgan Kaufmann / Elsevier (2005)
31. Huberman, B.A., Lukose, R.M., Hogg, T.: An economics approach to hard computational problems. Sci. 275, 51–54 (1997)
32. Hutter, F., Hamadi, Y., Hoos, H.H., Leyton-Brown, K.: Performance prediction and automated tuning of randomized and parametric algorithms. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 213–228. Springer, Heidelberg (2006)

33. Hutter, F., Hoos, H.H., Stützle, T.: Automatic algorithm configuration based on local search. In: AAAI, pp. 1152–1157. AAAI Press, Menlo Park (2007)
34. Jakob, W.: HyGLEAM – an approach to generally applicable hybridization of evolutionary algorithms. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 527–536. Springer, Heidelberg (2002)
35. Jakob, W.: Towards an adaptive multimeme algorithm for parameter optimisation suiting the engineers' needs. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 132–141. Springer, Heidelberg (2006)
36. Johnson, D., Demers, A., Ullman, J., Garey, M., Graham, R.: Worst-case performance bounds for simple one-dimensional packaging algorithms. SIAM J. on Comput. 3(4), 299–325 (1974)
37. Keller, R.E., Poli, R.: Linear genetic programming of parsimonious metaheuristics. In: Srinivasan, D., Wang, L. (eds.) 2007 IEEE Congress on Evolutionary Computation, pp. 4508–4515. IEEE Computational Intelligence Society/IEEE Press, Singapore (2007)
38. Koza, J.R.: Genetic Programming: on the Programming of Computers by Means of Natural Selection. The MIT Press, Boston (1992)
39. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. The MIT Press, Cambridge (1994)
40. Koza, J.R., Bennett III, F.H., Andre, D., Keane, M.A.: Genetic Programming III: Darwinian Invention and Problem solving. Morgan Kaufmann, San Francisco (1999)
41. Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence (Genetic Programming). Springer, Heidelberg (2003)
42. Koza, J.R., Poli, R.: Genetic programming. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, pp. 127–164. Springer, Boston (2005)
43. Krasnogor, N., Gustafson, S.: A study on the use of 'self-generation' in memetic algorithms. Nat. Comput. 3(1), 53–76 (2004)
44. Krasnogor, N., Smith, J.E.: Emergence of profitable search strategies based on a simple inheritance mechanism. In: Proceedings of the 2001 Genetic and Evolutionary Computation Conference, pp. 432–439. Morgan Kaufmann, San Francisco (2001)
45. Mockus, J.: Application of bayesian approach to numerical methods of global and stochastic optimization. J. of Glob. Optim. 4(4), 347–366 (1994)
46. Nareyek, A.: Choosing search heuristics by non-stationary reinforcement learning. In: Resende, M.G.C., de Sousa, J.P. (eds.) Metaheuristics: Computer Decision-Making, ch. 9, pp. 523–544. Kluwer, Dordrecht (2003)
47. Ong, Y.S., Keane, A.J.: Meta-lamarckian learning in memetic algorithms. IEEE Trans. on Evol. Comput. 8, 99–110 (2004)
48. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of adaptive memetic algorithms: a comparative study. IEEE Trans. on Syst. Man and Cybern. Part B 36(1), 141–152 (2006)
49. Ozcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive survey of hyperheuristics. Intell. Data Anal. 12(1), 3–23 (2008)
50. Poli, W.B.R., Langdon, N.F.M.: A Field Guide to Genetic Programming. Lulu Enterprises, UK (2008)
51. Rhee, W.T., Talagrand, M.: On line bin packing with items of random size. Math. Oper. Res. 18, 438–445 (1993)

52. Ropke, S., Pisinger, D.: A unified heuristic for a large class of vehicle routing problems with backhauls. Eur. J. of Oper. Res. 171(3), 750–775 (2006)
53. Ross, P.: Hyper-heuristics. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, ch. 17, pp. 529–556. Springer, Heidelberg (2005)
54. Ross, P., Marin-Blazquez, J.G., Schulenburg, S., Hart, E.: Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2003, pp. 1295–1306. Springer, Heidelberg (2003)
55. Seiden, S.S.: On the online bin packing problem. J. ACM 49(5), 640–671 (2002)
56. Smith, J.E.: Co-evolving memetic algorithms: A review and progress report. IEEE Trans. in Syst. Man and Cybern. Part B 37(1), 6–17 (2007)
57. Stephenson, M., O'Reilly, U., Martin, M., Amarasinghe, S.: Genetic programming applied to compiler heuristic optimisation. In: Proceedings of the Eur. Conference on Genetic Programming, pp. 245–257. Springer, Heidelberg (2003)
58. Terashima-Marin, H., Flores-Alvarez, E.J., Ross, P.: Hyper-heuristics and classifier systems for solving 2D-regular cutting stock problems. In: Beyer, H.G., O'Reilly, U.M. (eds.) Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2005, Washington DC, USA, June 25-29, pp. 637–643. ACM, New York (2005)
59. Terashima-Marin, H., Ross, P., Valenzuela-Rendon, M.: Evolution of constraint satisfaction strategies in examination timetabling. In: Proc. of the Genetic and Evolutionary Computation Conf. GECCO 1999, pp. 635–642. Morgan Kaufmann, San Francisco (1999)
60. Thrun, S., Pratt, L.: Learning to learn: Introduction and overview. In: Thrun, S., Pratt, L. (eds.) Learning To Learn. Kluwer Academic Publishers, Dordrecht (1998)
61. Vazquez-Rodriguez, J.A., Petrovic, S., Salhi, A.: A combined meta-heuristic with hyper-heuristic approach to the scheduling of the hybrid flow shop with sequence dependent setup times and uniform machines. In: Proceedings of the 3rd Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2007), pp. 506–513 (2007)
62. Wah, B.W., Ieumwananonthachai, A.: Teacher: A genetics-based system for learning and for generalizing heuristics. In: Yao, X. (ed.) Evol. Comput., pp. 124–170. World Scientific Publishing Co. Pte. Ltd, Singapore (1999)
63. Wah, B.W., Ieumwananonthachai, A., Chu, L.C., Aizawa, A.: Genetics-based learning of new heuristics: Rational scheduling of experiments and generalization. IEEE Trans. on Knowl. and Data Eng. 7(5), 763–785 (1995)

Adaptive Constraint Satisfaction: The Quickest First Principle

James E. Borrett and Edward P.K. Tsang

Abstract. The choice of a particular algorithm for solving a given class of constraint satisfaction problems is often confused by exceptional behaviour of algorithms. One method of reducing the impact of this exceptional behaviour is to adopt an adaptive philosophy to constraint satisfaction problem solving. In this report we describe one such adaptive algorithm, based on the principle of chaining. It is designed to avoid the phenomenon of exceptionally hard problem instances. Our algorithm shows how the speed of more naïve algorithms can be utilised safe in the knowledge that the exceptional behaviour can be bounded. Our work clearly demonstrates the potential benefits of the adaptive approach and opens a new front of research for the constraint satisfaction community.

1 Introduction

Constraint Satisfaction Problems occur in many areas of everyday life. These range from problems such as timetabling and transport planning to configuration problems and document layout design. In all cases, the notion of a Constraint Satisfaction Problem (CSP) is characterised by the need to assign values to elements of the problem instances, these values coming from a finite set of possibilities and subject to a set of rules or constraints [31] [23].

Once a CSP has been identified there are whole host of problem solving techniques which have been developed for solving them [23]. The most basic of these is the simple backtracking algorithm but more sophisticated algorithms such as look-ahead approaches have been shown to be highly effective [13] and are commonly used in commercial software libraries such as ILOG Solver [22]. Heuristic search has been applied to CSP with success, e.g. see [14] [18] [32], and have also been embedded in industrial packages such as iOpt [33].

James E. Borrett and Edward P.K. Tsang
University of Essex, Dept. of Computer Science, Wivenhoe Park, Colchester CO4 3SQ, United Kingdom
e-mail: {jborrett,edward}@essex.ac.uk

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 203–230.
springerlink.com

More formally, the constraint satisfaction problem (CSP) can be defined in terms of the triple <Z, D, C>, where Z is a set of variables, D is a mapping of the variables in Z to domains and C is a set of constraints [31]. Given this definition of a CSP, there are many ways in which different types of problem can be classified, in terms of the elements of Z, D and C[1]. This classification may then be used as a basis for the selection of a particular algorithm to solve that class of problems.

There is, however, a significant complication with the definition of CSP classes. Sometimes particular instances of problems in a class may exhibit exceptional qualities, in terms of the solving abilities of the chosen algorithm. One clear example of this is the phenomenon of exceptionally hard problem instances [28], or EHPs as they shall be referred to in this paper.

The example of EHPs is illustrative of the dilemma posed to the problem solver. There is a clear choice of either using a naïve algorithm which is likely to solve most instances very quickly, at the risk of catastrophic encounter with an EHP, or to choose a more complex algorithm, which has a far lower probability of encountering EHPs[2]. However, as is often the case, the use of more complex algorithms entails an overhead.[3]

One approach which can overcome this dilemma is to use a more flexible approach which we describe as adaptive constraint satisfaction. The notion of adaptive constraint satisfaction can be encapsulated in the following description:

> Adaptive Constraint Satisfaction is a general philosophy for solving constraint satisfaction problems. It aims to make use of the many algorithms and techniques available by relaxing the commitment to a single algorithm when solving a particular CSP, allowing for the active modification or switching of algorithms and models during the search process.

Built upon the Adaptive Constraint Satisfaction context was a set of research projects[4]. Adaptive constraint satisfaction is based on the belief that there is no "best algorithm" in constraint satisfaction – different algorithms work for different problem instances – an idea that was later articulated as the "No Free Lunch Theorem" [35, 36][5]. Based on this belief, Kwan et al [17] [30] developed a machine learning framework for learned mappings from CSPs to algorithms and heuristics. Given

[1] In [3] the issue of classifying different formulations of the same problem is considered.

[2] In the context of complete algorithms, [25], [26] suggest it is likely that investing in more complex algorithms, such as forward checking with conflict-directed backjumping [20], will decrease the frequency of encounters with EHPs.

[3] Given that EHPs are algorithm dependent, as explained above, another approach is to restart the search with, say a random algorithm. The difficulty in this approach is deciding when to restart. Abandoning the search prematurely means a waste of search effort; if one is not careful, one could end up restarting indefinitely. That motivated us to develop a mechanism to recognize thrashing.

[4] See Adaptive Constraint Satisfaction Project (1992-98) http://csp.bracil.net/acs.html

[5] According to this theorem "there is no free lunch when the probability distribution on problem instances is such that all problem solvers have identically distributed results". See Wikipedia http://en.wikipedia.org/wiki/No_free_lunch_in_search_and_optimization (accessed 18 August 2008)

a CSP, the algorithm picked may not work efficiently. This is because such mappings were generated statistically, which may not apply to every problem instance. The problem instance on hand may be "exceptionally hard" to the chosen algorithm and heuristic. Therefore, part of the Adaptive Constraint Satisfaction project was to develop measures for monitoring algorithms when they search. Every algorithm is designed to exploit certain characteristics of the problem instance. If an algorithm/heuristic does not do what it is supposed to do, it should be stopped, and a different algorithm/heuristic should be used. For example, lookahead algorithms [13] are designed to propagate constraints in order to prune the search space and detect dead-ends. If, during the search, it is found that not much of the search space is pruned, and a large amount of constraint propagation effort has resulted in few dead-ends being detected, the lookahead algorithm that is currently used should be replaced.

In this paper, we outline a particular instance of the adaptive approach where we make use of Algorithmic Chaining. The result is REBA (for Reduced Exceptional Behaviour Algorithm) which is designed to avoid the phenomenon of exceptionally hard problems in the so called easy region for solvable CSPs. REBA operates on complete search methods – methods that explore the search space systematically and entirely if necessary.

2 The Adaptive Strategy

We have defined adaptive constraint satisfaction as a general approach to solving CSPs. Within that approach there are many possible strategies. We examine one particular adaptive strategy, designed to reduce the significance of EHPs by utilising algorithmic chaining. Algorithmic chaining uses a set of algorithms, arranged in a pre-determined order, combined with a switching mechanism. The switching mechanism monitors the search process of the current algorithm and, should certain conditions occur, stops the current algorithm, trying again with the next algorithm in the chain. In this section we discuss these two elements of the strategy.

2.1 Chain Design

As noted in [25], [26] the phenomenon of EHPs appears to affect different algorithms to different degrees. However, the trend tends to be for more naïve algorithms, such as simple chronological backtracking algorithms, to be more susceptible. This presents us with two potentially useful measures for ranking algorithms. The first is the cost to solve 'normal' occurrences of CSPs (measured by the median cost), and the second is the algorithms sensitivity to EHPs. An example of possible differences in ranking is given in Table 1.

If we can determine similar rankings to those in Table 1, we would have enough information to design a useful chain for solving CSPs in the easy region whilst increasing the likelihood of avoiding the potentially catastrophic effects of

Table 1 Example showing how the ranking of algorithms can differ when based on median cost of solving CSPs, and sensitivity to EHPs

Rank	Algorithm Complexity	Median Cost	Sensitivity to EHPs
1	X	X	Z
2	Y	Y	Y
3	Z	Z	X

encountering an EHP. The chain can simply be set to an ordering based on the "Quickest First Principle" (QFP), where quickest indicates the algorithm with the best median performance.

We wanted to design an algorithm for solving easy solvable problems without failing in EHPs. Using QFP means that we always have the potential for solving the CSPs quickly. However, if we can detect that the current algorithm is not working well, we could switch to the next quickest algorithm, and so on. As a result we can still benefit from the speed of the naïve algorithms while at the same time having the capability to resort to more complex algorithms in the event that a switch scenario is detected.

While there is some overhead involved in this approach, the benefits can be considerable. For example, the ability to use a simple algorithm can result in an order of magnitude gain in performance over its more complex counterparts. Another advantage is that in the event of a bad initial choice of algorithm, we are not stuck with it. Mistakes of this nature will be rectified when we switch away from the bad choice.

2.2 Switching Policy

The main requirements of the switching mechanism are that it can detect the phenomena you wish to avoid, while adding only minimal overheads to the basic algorithm. For REBA this means we need to predict the thrashing type behaviour associated with EHPs encountered by naïve algorithms, using a simple and efficient prediction method.

There appear to be many types of thrashing in CSPs. [25], [26] note the basic thrashing scenario is often seen in chronological backtracking algorithms such as forward checking [13]. This is the worst type of thrashing, where the algorithm visits all nodes in a sub-tree of the search space when it is futile to do so. It is not experienced by more complex algorithms, such as intelligent backjumping algorithms. However the idea of a search sub-space being repeatedly visited when it is futile to do so still occurs in these algorithms, the main difference being the amount of the sub-space visited.

At the heart of the switching mechanism of REBA is the Monitor Search Level (MSL) thrashing predictor which is described in detail in Section 3.2. MSL represents one possible mechanism which attempts to predict when thrashing type behaviour is likely to occur such that only a small portion of any futile sub search space is actually explored by the algorithm in question. Using a sensitivity threshold supplied to it, the predictor will suggest that a switch is necessary if the threshold is reached.

3 The Reduced Exceptional Behaviour Algorithm (REBA)

Having outlined the basic strategy for our Reduced Exceptional Behaviour Algorithm, we give more details of its design. We also give a description of the prediction mechanism used by REBA.

3.1 The REBA Algorithm Chain

The chain used by REBA is designed using the principles outlined in Section 2. This chain uses a selection of algorithms with good median performance on easy soluble CSPs, and a selection of algorithms with good worst case performance. These cover a range of complete search techniques including features such as forward checking, backjumping and heuristics which cover both static and dynamic variable orderings. No stochastic algorithms are considered for REBA, but this should not rule out the possibility of using them in alternative adaptive approaches. Space would not allow us to go into details of these algorithms. Relevant pointers are provided here. [31] explains most of these algorithms. In a way, it is not essential to understand details of these algorithms. For this paper, the relevant point is that they cover a wide range of algorithms and heuristics with diversified strength.

Having carried out some preliminary investigations, we chose to use the following algorithms;

BM+MWO	back-marking [9] with the minimum width ordering heuristic [6]
BMCBJ+MWO	back-marking with conflict-directed backjumping [20] with the minimum width ordering heuristic
BMCBJ+MDO	back-marking with conflict-directed backjumping [20] with the maximum degree ordering heuristic[6]
FCCBJ+BZ	forward checking with conflict-directed backjumping [20] with the Brélaz ordering heuristic [29], [4]
MAC+MDO	Maintain Arc Consistency [24] with the maximum degree ordering heuristic

We propose to use these algorithms in the following chain to tackle problem instances in the easy, soluble region:

BM+MWO → BMCBJ+MWO → BMCBJ+MDO → FCCBJ+BZ → MAC+MDO

The reasoning behind this chain is that BM+MWO is very fast for many easy soluble problem instances, but very susceptible to EHPs. However, it might succeed in a very quick solution, otherwise thrashing will be detected. In the event that

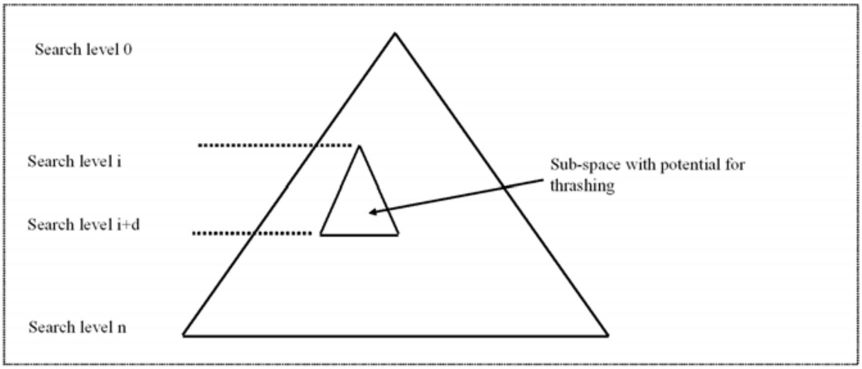


Fig. 1 An example of a sub search space

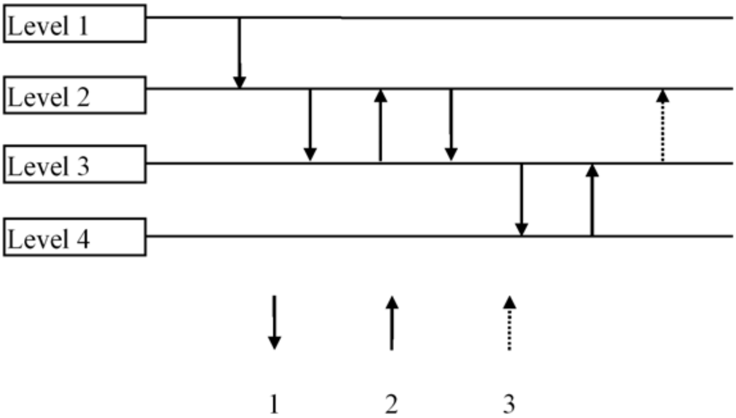


Fig. 2 The types of progress during search (see text for explanations)

BM+MWO fails, we try adding intelligent backjumping to it. If this fails, we try changing the ordering, since a bad ordering is often a contributing factor to EHPs [28]. If these simpler algorithms fall victim to an EHP, we attempt to use a form of forward checking with conflict-directed backjumping and a dynamic variable ordering. Finally, if this fails, we resort to another algorithm which has relatively low susceptibility to EHPs, MAC+MDO.

3.2 The Monitor Search Level (MSL) Thrashing Predictor

In this section we describe the Monitor Search Level (MSL) thrashing predictor. We describe the behaviour MSL watches for, and explain how it decides when this behaviour is sufficiently clear for thrashing to be predicted.

As a basis for the design of MSL we defined the following functional specification;

> Given a CSP, an algorithm, and a variable ordering, the predictor should monitor the progress of the search and be able to predict if thrashing is likely to occur during the search.

One indication of thrashing is when the search from a particular level i never proceeds beyond a certain depth, d, and that a large proportion of the search space between level i and level $i+d$ is explored (i.e. little pruning takes place between these two levels, see Figure 1). Such a situation can occur when the culprits of the failure at level $i+d$ precede the level i. MSL is a simple method which uses this observation to predict thrashing type behaviour.

Before discussing MSL in more detail, we must identify three distinct types of progress which occur during search. These are presented in figure 2.The types of progress are defined as;

1. A value is found for the current variable which is compatible with all previous assignments, or future variables in the case of lookahead algorithms. For example the second arrow in Figure 2, where a value is found for the variable at level 2 which is compatible with the value assigned to the variable at level 1.
2. Backtracking occurs after finding no values for the current variable which are compatible with previous assignments, or future variables in the case of lookahead algorithms. For example the third arrow in Figure 2, where no value can be found for the variable at level 3 which is compatible with the current assignments of the variables at levels 1 and 2. This will be known as a No Assigned Value (NAV) backtrack. The NAV backtrack occurs at the tail of the arrow, level 3. At the head of the arrow, level 2 learns of an Unsuccessful Subspace Search (USS).
3. Backtracking occurs, but only after at least one value has been found for the current variable which is compatible with the assignments of previous variables, or future variables in the case of lookahead algorithms (Meaning the search must have progressed at least one level further down than the current one). For example the seventh arrow in Figure 2, where a value for the variable at level 3 has been found which is compatible with the assignments of the variables at levels 1 and 2, but is later rejected because no value can be found for the variable at level 4. This will be known as a Successfully Assigned Values (SAV) backtrack. The SAV backtrack occurs at the tail of the arrow, level 3. At the head of the arrow, level 2 learns of a USS.

During the search MSL keeps track of the last level at which a NAV backtrack occurred. This is considered to be the deepest level of the current search sub-space. We will refer to this level as DEEPEST.

In addition, for each level in the search, MSL keeps track of two values. Firstly a count indicating the number of USS's which returned to the level with the same value for DEEPEST. Secondly a record of the value of DEEPEST when this count

Table 2 Possible actions of MSL on count$_i$ and DL$_i$ for level i

	(1) DEEPEST < DL$_i$	(2) DEEPEST = DL$_i$	(3) DEEPEST > DL$_i$
(a) USS	No action	Increase count$_i$ by 1; Check count against threshold	Set count$_i$ to 1; Set DL$_i$ to DEEPEST
(b) NAV Backtrack	Set DEEPEST to i	Set DEEPEST to i	Not Possible
(c) SAV Backtrack	Reset count$_i$ to 0; Set DL$_i$ to DEEPEST	No action	Not Possible

is started. We will refer to these values as count$_i$ and DL$_i$ respectively, where i is the level they refer to.

In considering how the count is maintained, we must examine the seven possible cases. These depend on whether a USS, a NAV backtrack or a SAV backtrack is occurring, and what the value of DEEPEST is compared to the value of DL$_i$ for the level. Table 2 illustrates the different actions taken at a given level, i, depending on these circumstances.

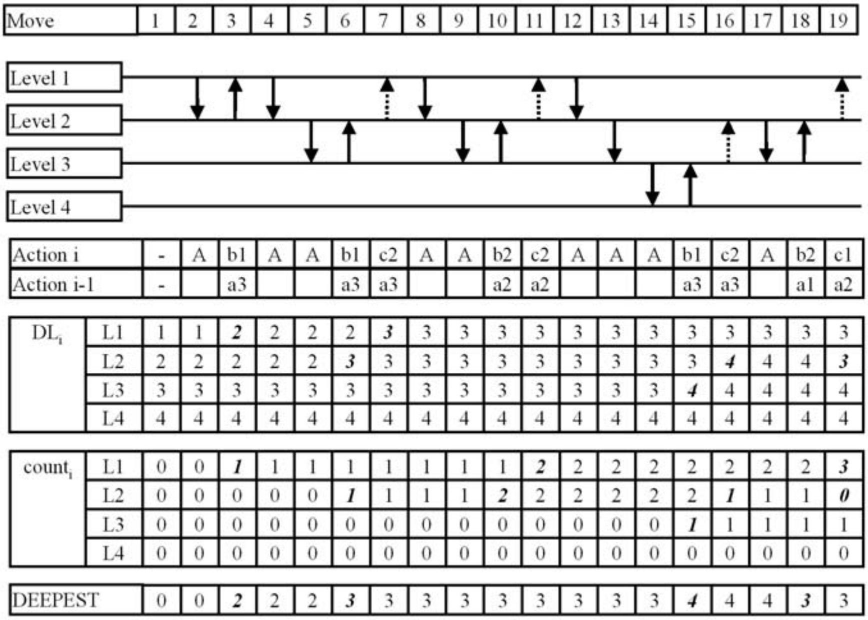
Some points should be noted here:

- DEEPEST and count$_i$ are initialised to 0 and DL$_i$ are initialised to i
- DEEPEST can only be changed by a NAV backtrack occurring, and always changes when such a backtrack occurs.

Figure 3 gives an example illustrating the possible situations encountered by MSL. Each column in Figure 3 represents either an assignment, a NAV backtrack, or a SAV backtrack together with a USS if applicable (with the exception of the first column). The numbers below the arrow indicate the values of DL$_1$,..,DL$_4$, count$_1$,...,count$_4$ and DEEPEST **after** the actions for that column have been carried out. The values of the actions indicate which entries in Table 2 apply to the above arrow[7]. This includes actions at both the tail and the head of the arrow. The first column simply shows the initial values before the search begins.

As an example consider columns 14 to 16. Column 14 shows a simple assignment to the variable at level 3, action A. No further actions take place. Column 15 then shows a NAV backtrack from the variable at level 4. When the backtrack occurs, DL$_4$ = 4 and DEEPEST = 3, so DL$_4$ > DEEPEST and entry b1 in Table 2 applies to level 4. As a result DEEPEST is set to the value of i, i.e. DEEPEST = 4. At the

[7] The entry A indicates a successful assignment, no action is taken.



Move		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Action i		-	A	b1	A	A	b1	c2	A	A	b2	c2	A	A	A	b1	c2	A	b2	c1
Action i-1		-		a3			a3	a3			a2	a2				a3	a3		a1	a2
DL_i	L1	1	1	***2***	2	2	2	***3***	3	3	3	3	3	3	3	3	3	3	3	3
	L2	2	2	2	2	2	***3***	3	3	3	3	3	3	3	3	3	***4***	4	4	***3***
	L3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	***4***	4	4	4	4
	L4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
$count_i$	L1	0	0	***1***	1	1	1	1	1	1	1	***2***	2	2	2	2	2	2	2	***3***
	L2	0	0	0	0	0	***1***	1	1	1	***2***	2	2	2	2	2	***1***	1	1	***0***
	L3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	***1***	1	1	1	1
	L4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DEEPEST		0	0	***2***	2	2	***3***	3	3	3	3	3	3	3	3	***4***	4	4	***3***	3

Fig. 3 Example search

head of the arrow USS entry *a*3 applies (because DEEPEST = 4 and DL_3 = 3) and $count_3$ is set to 1 with DL_3 being set to DEEPEST.

Column 16 shows a SAV backtrack from the variable at level 3. When the backtrack occurs, DL_3 = 4 and DEEPEST = 4. Since DL_3 = DEEPEST entry *c*2 in Table 2 applies and no action is taken at level 3. At the head of the arrow USS entry *a*3 applies and $count_2$ is set to 1 with DL_2 being set to DEEPEST.

3.2.1 Effectiveness of Thrashing Prediction Mechanisms

Having defined the function of our prediction mechanism, we also define a set of criteria for evaluating its effectiveness. These criteria are based on three main functions;

i It should predict as exceptionally hard those problem instances with high search cost for the current algorithm.
ii The computational cost of predicting a CSP to be exceptionally hard should be low and preferably not exceed the median cost. It should also be cheap in terms of space.
iii It should not be so sensitive that too many problem instances are predicted to be exceptionally hard. A high proportion of the problem instances with search

costs of median or lower should not be predicted to be exceptionally hard for the current algorithm.

3.3 The REBA Switching Mechanism

The MSL predictor is used by REBA for its switching mechanism. This is done by REBA supplying the predictor with a formula for calculating the threshold. If the threshold is exceeded, then MSL suggests that a switch should take place. As a result, REBA will switch to the next algorithm in the chain.

We have experimented with a threshold based on the domain size of the variables, and the number of levels separating the current level *i* and DL_i. The base threshold is a multiple of the domain size. The number of separating levels is taken as DL_i - *i*. The more separating levels, the lower the threshold has to be for switching to occur. The formula used is;

$$Threshold = base * \left(\frac{n - separation}{n}\right)$$

where: - base is the base threshold, which is a linear function of the domain size
- *n* is the number of variables,
- *separation* is the number of separating levels (DL_i- i).

The threshold is adjusted according to *separation* to improve the sensitivity of detection when the subspace is only searched sparsely, as might be the case with intelligent backjumping algorithms.

Note that in subsequent experiments a suffix is given to the name of REBA. This suffix indicates the multiples of the domain size used for the base threshold.

4 Experiments

In order to evaluate the overall performance of REBA and the effectiveness of its switching mechanism we carried out an experiment on different classes of easy soluble CSPs (which is what REBA is designed to tackle). This section describes details of our experiment as well as presenting our results.

4.1 Experimental Design

The main aim of our experiment was to compare the performance of REBA with two types of algorithms - those exhibiting good median performance in the easy soluble region, and those that have a good worst case performance on easy soluble region. Randomly generated CSPs are used to evaluate REBA. They allow us to control the tightness of problem classes, and therefore select appropriate problem classes for experimentation.

The actual CSPs we used were based on randomly generated binary CSPs classified by the tuple <n, m, p1, p2>, where the elements of the tuple are defined as;

n number of variables
m uniform domain size
$p1$ density of constraints in the constraint graph
$p2$ tightness of individual constraints[8] i.e. the percentage of incompatible assignments between the two variables involved in the constraint

Specifically, we wanted to conduct our experiments on problems in the so-called easy soluble region where exceptionally hard problem instances were likely to occur. As a result, we chose the class <50, 10 , 0.1, 0.35 - 0.5 >. This range of $p2$ gives us a spread of problem instances in the region of interest and it also includes some of the sets of problems used in [25] and [26], where EHPs were investigated.

The algorithms we chose for comparison, based on initial tests of problem instances in the class description above, were as follows;

BMCBJ+MWO back-marking with conflict-directed backjumping with the static minimum width ordering - this combination gives a low median performance but has a sensitive worst case performance in the region of interest.

FCCBJ+BZ forward checking with conflict-directed backjumping with the dynamic Brélaz ordering - this combination gives a relatively high median performance but a good worst case performance in the region of interest.

MAC+MDO maintain arc-consistency with the static maximum degree ordering - this combination also gives a relatively high median performance but a good worst case performance in the region of interest.

The CSPs for our experiments were generated at intervals of $p2$ of 0.01 and the sample size for each data point was 1000. In order to limit the impact of EHPs on our experimentation time, we limited the actual process CPU time for any given run to 30 minutes. Where this time is exceeded, the compatibility check count up to that time was recorded[9].

The effect of using such a limit is that for a few data points, for the BMCBJ+MWO combination, the limit was reached. This does not detract from the essence of our results, however, since the effect of any EHP is still clearly visible. The truncated values are many orders of magnitude above the median search cost.

[9] Note that the algorithms were implemented in C++ and run on DEC Alpha 3000 Model 600 AXP workstations running at 175 MHz.

4.2 The Effectiveness of REBA

The results of our experiment in measuring the effectiveness of REBA are presented in Figures 4–7[10]. The results clearly show that the use of algorithmic chaining in REBA has produced a good worst case performance where the impact of EHPs has been significantly reduced. This is evident in the worst case plots of Figures 5 and 7. REBA even outperforms FCCBJ+BZ in many cases. At the same time, the median

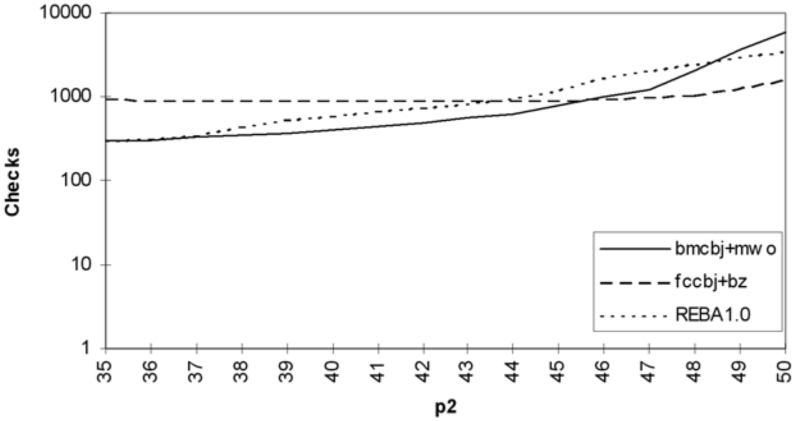


Fig. 4 Median performance on 50 variable problems in terms of compatibility checks

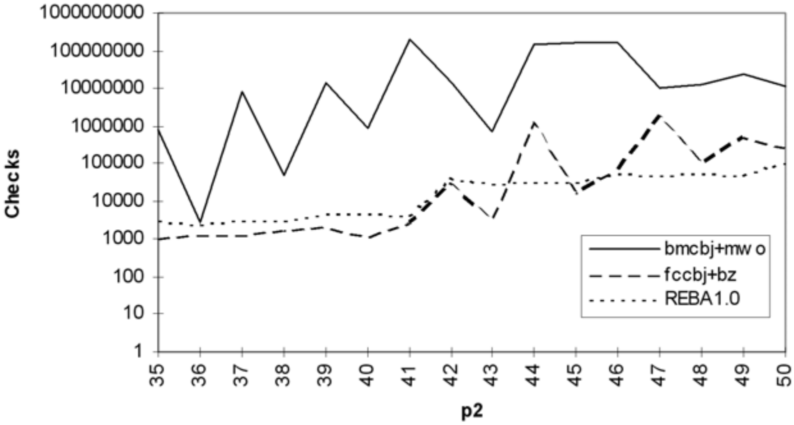


Fig. 5 Worst case performance on 50 variable problems in terms of compatibility checks

[10] We only present cpu time results for MAC since our implementation is the same as that of [24] where the compatibility check count is not a true reflection of the work done by MAC.

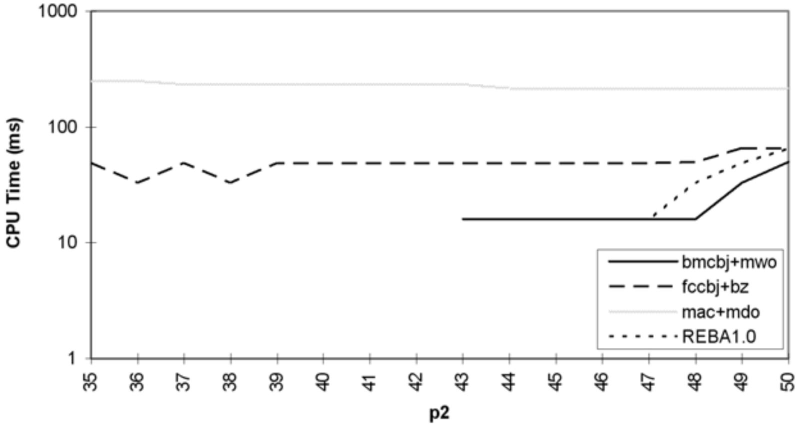


Fig. 6 Median performance on 50 variable problems in terms of cpu time. (Note that where the plot for REBA and BMCBJ+MWO does not exist this means the median time was less that one clock cycle and hence does not show in the logarithmic scale)

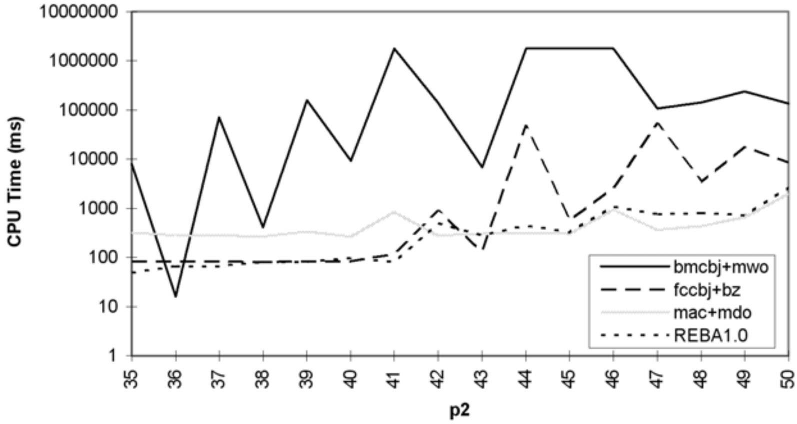


Fig. 7 Worst case performance on 50 variable problems in terms of cpu time

performance of REBA is much better than that of the more complex algorithms, in most cases. This is particularly apparent when the CPU time is considered as in Figures 6 and 7.

It should be noted that we have tested REBA on problems in the easy region. This is because we advocate that different types of problem would be tackled by different algorithms as noted in [30]. REBA, by design, appears to be useful in tackling problems in the easy region on the soluble side of the phase transition. It is the subject of further work to investigate the applicability of the strategies used in REBA to tackling other problem types such as those in the phase transition.

4.3 Evaluation of the MSL Predictor

To see how effective the switch detection mechanism in REBA is, we carried out a further experiment. This time, we did not run the chain of algorithms. Instead, we ran a version of BM+MWO, which included the MSL predictor, and monitored where a switch was predicted (if one was required). If a switch was predicted, the number of compatibility checks was recorded and the algorithm was allowed to continue running to completion to see what the actual outcome would have been[11]. We also repeated this experiment for an intelligent backjumping algorithm, BM-CBJ+MWO, allowing us to observe the effectiveness of MSL in these two types of algorithm.

For the BM+MWO combination, a problem set of 1000 CSPs were generated with the specification $<50, 10, 0.1, 0.4>$. For the BMCBJ+MWO combination 1000 CSPs with the specification $<50, 10, 0.1, 0.5>$. This difference in $p2$ is a reflection of the location where REBA was observed to have switched from these algorithms in the experiment detailed in Section 4.2.

In Section 3 we defined three criteria for a evaluating a thrashing prediction mechanism. We present our results in three ways to address these criteria. In Figures 8 and 9 we see how effective MSL is at filtering out problems where the actual cost of search to completion would have been high, including the possibility of EHPs. These histograms show the actual cost to completion of all the instances where a switch would have taken place[12](of which there were 589 for BM+MWO and 693 for BMCBJ+MWO).

These two figures show how there are many high cost searches predicted by MSL to be thrashing.

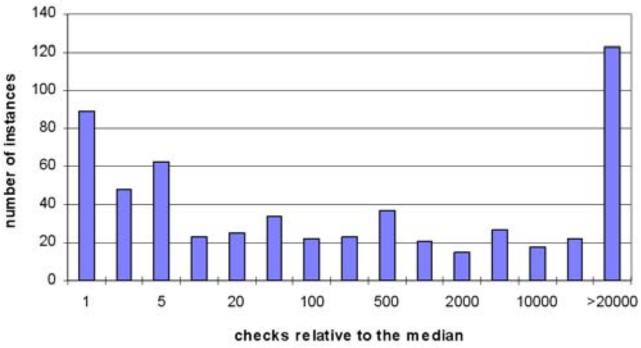


Fig. 8 Ultimate search cost for BM+MWO had a switch not been predicted (total of 589 instances)

[11] For the purposes of this experiment we used a base threshold equal to the domain size of the variables.

[12] The results are presented as multiples of the median search cost when considering the cost to completion for all CSPs in the sample of 1000.

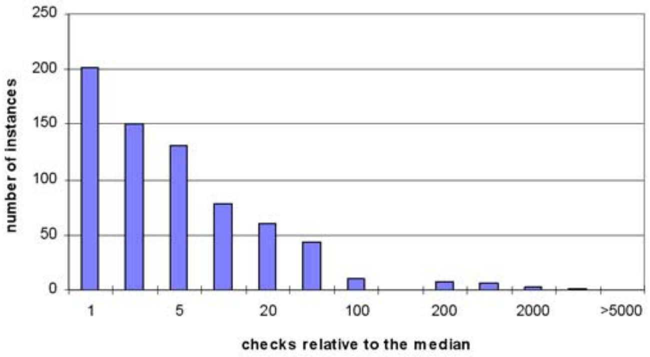


Fig. 9 Ultimate search cost for BMCBJ+MWO had a switch not been predicted (total of 693 instances)

The second criterion was that the cost to detection should be low. Figures 10 and 11 show the actual search cost up to detection for the instances where a switch was suggested.

As can be seen from these figures the performance is good, since the median cost for predicting a switch in BM+MWO was always less than the median search cost when all CSPs are considered. For BMCBJ+MWO a similar result can be seen, with the exception of a few cases. However, even with these exceptions, there are no cases where the cost exceeds five times the overall median.

Finally, the third criterion was that the prediction mechanism should not be too sensitive and prevent completion of search for the many problem instances that would have only had median cost to solve to completion. Figures 12 and 13 show the cost of search for all the problem instances where no switch was predicted place (of which there were 411 for BM+MWO and 307 for BMCBJ+MWO).

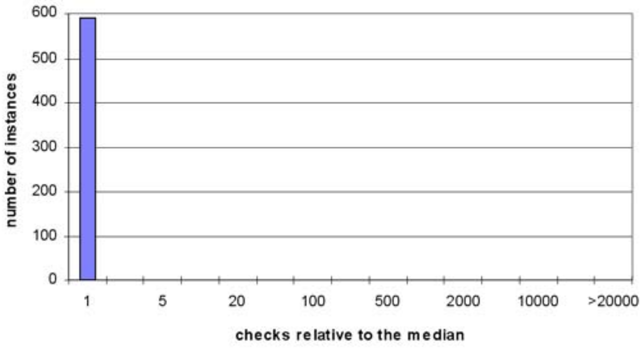


Fig. 10 Cost to predict a switch for BM+MWO (total of 589 instances)

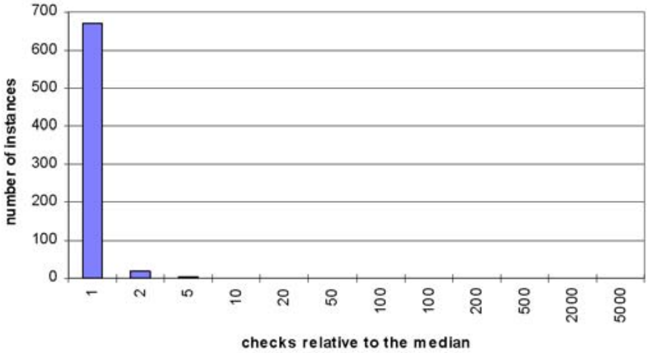


Fig. 11 Cost to predict a switch for BMCBJ+MWO (total of 693 instances)

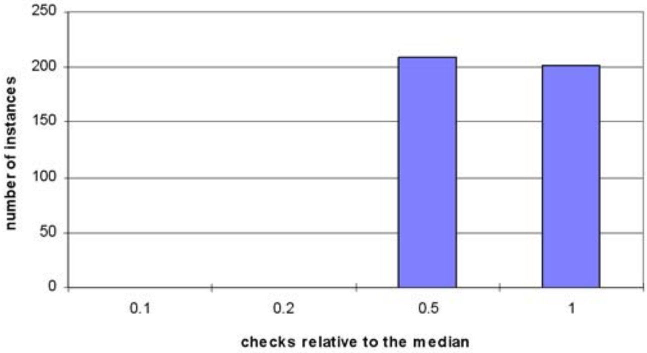


Fig. 12 Search cost for problems where no switch was predicted for BM+MWO (total of 411 instances)

This clearly shows that no high cost problem instances are allowed through and that there were many low cost problems let through. For BM+MWO, the maximum search cost for a CSP in this set was less than the median for all problems. In the case of BMCBJ+MWO, the maximum never exceeds five times the median.

From Figures 9–14 it is clear that the MSL predictor used for this version of REBA, with a base threshold of 1.0, has performed very effectively, and that the criteria laid out in Section 3.2 are largely fulfilled.

There is obviously a trade off when choosing the value for the threshold such that no exceptionally hard problems are encountered, whilst at the same time allowing the majority of the easier problems to be solved. The base threshold we have used was equal to the domain size of the variables and was the same for all algorithms. However, it may be possible to improve the effectiveness of algorithms such as REBA by using a different threshold, or perhaps by using different thresholds for the different algorithms in the chain.

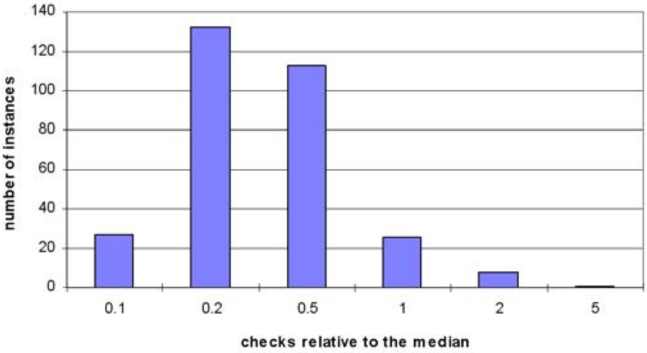


Fig. 13 Search cost for problems where no switch was predicted for BMCBJ+MWO (total of 307 instances)

We have experimented with different thresholds and find that they also produce good results when compared to the algorithms used in the above tests. We have also looked at how REBA performs with larger problem sizes. Again, REBA performs well. These results are given in Section A2.

5 Discussion

In this chapter we have demonstrated the potential of adaptive constraint satisfaction. We have outlined a particular application of the adaptive approach using the technique known as algorithmic chaining. This technique was incorporated in an algorithm that we have named REBA, and has been shown to be effective in reducing susceptibility to exceptionally hard problem instances.

The REBA algorithm makes use of a mechanism for predicting when thrashing type behaviour is likely to occur. This notion of prediction is one of the keys to the adaptive approach since it is prediction that allows algorithms to avoid search spaces before they can impact significantly on the overall search. The MSL mechanism used here is computationally very cheap and it has been shown to be reasonably accurate.

Experiments with the REBA algorithm, which is specifically designed to reduce the impact of exceptionally hard problem instances, show that it is possible to take advantage of the speed of basic constraint satisfaction algorithms when solving easy, soluble problem instances, while at the same time allowing us to bound the exceptional behaviour of these algorithms when exceptional problem instances are encountered. The principle of using the quickest algorithm first means that the best case performance of the naïve algorithms always has a chance of being achieved. It also gives the opportunity for fast solutions to be provided in the event that “exceptionally easy” problem instances are encountered - this could be significant if a similar method were to be used on, for example, hard classes of CSPs.

REBA represents a novel approach which demonstrates the potential of collaboration between algorithms. Exceptionally hard problem instances are so punitive in terms of cost that effective detection of potential traps for naïve algorithms using a low cost detection mechanism such as MSL that it is possible to make use of more sophisticated algorithms when they are needed, without incurring their general overhead.

REBA demonstrates a number of key ideas of adaptive constraint satisfaction. First, it recognizes that no algorithm is best for every CSP, as the No Free Lunch Theorem does. It also demonstrates that search performance could be monitored to see whether the algorithm is achieving what it is designed to achieve. (In fact, there is no reason why algorithms should not be monitored beyond the constraint satisfaction context.) REBA also demonstrates that efficiency can be gained by a rigid chain of algorithms.

Gomes et al [12] studied when expensive search happens in a given algorithm, which is highly related to thrashing detection in REBA. Dynamic restart was also investigated by a number of other works. Kautz et al [38] defined a set of policies for restarting the search. Gagliolo and Schmidhuber [8, 7] proposed to model the runtime distribution – if training is possible – and use the estimated runtime distribution to decide when to restart. In the Solution-Guided Search algorithm, Beck [2] set, before the search starts, limits on the number of fails that the search is allowed to encounter. One could imagine using REBA's thrashing predictor to dynamically set this limit.

Using a portfolio of algorithms for constraint satisfaction has gained momentum in the last decade, see, for example, [15], [11] and [39]. Once a portfolio of algorithms is involved, selecting the right algorithm for the job becomes part of the research agenda in [10] and [39].

Many attempts have been made to learn from the problem solving experience. The idea of selecting the right heuristic algorithm during run time was developed by Allen and Minton [1]. Epstein et. al. [5] sought to learn search order heuristics during problem solving. Related to these ideas, Minton [37] demonstrated the possibility of synthesizing heuristics. Kern (2005) used population-based incremental learning to select algorithms and parameters. Kern's work is embedded in iOpt [33], which is used in many real-life dynamic problems, such as service scheduling [34].

This piece of work has opened many new areas of future work. One could further investigate the use of chains and similar methods of choosing appropriate algorithms to switch to in types of problems other than soluble easy CSPs. One could also look at other methods for detecting when it would be useful to switch between algorithms. This would involve identifying useful information that can be gathered during search. The actual process of switching could also be a source of improvement in efficiency. Ideally, information collected during the search could be used for selecting the new algorithm or heuristic. When switches take place, information gathered so far could be transferred to successive algorithms.

Acknowledgements. This work was supported by EPSRC research grant ref. GR/J/42878. The authors would like to thank Alvin Kwan for his useful comments on the contents of this paper. Natasha Walsh participated in the early part of this research. Christine Mumford (editor) and the anonymous referee provided us with insightful comments, which helped to improve the quality of this paper.

Please note: this paper is based on an extended form of Borrett, J., Tsang, E.P.K. & Walsh, N.R., Adaptive constraint satisfaction: the quickest first principle, Proceedings, 12th European Conference on AI, Budapest, Hungary, 1996, p.160-164.

Appendix

A.1 Tables of results for Figures 4–7

Table 3 Data for Figure 4, median performance on 50 variable problems in terms of compatibility checks

p2	bmcbj+mwo	fccbj+bz	REBA1.0
35	296	934	300
36	308	929	319
37	324.5	920	344
38	342	914	446
39	367	907.5	530
40	399.5	904	601
41	435.5	899	671
42	489.5	900	742.5
43	575	897	840
44	620	906	932
45	799	915	1216.5
46	1021.5	932	1691
47	1226	982	2037
48	2090	1064	2475.5
49	3624.5	1244	3002
50	5785	1628	3491.5

Table 4 Data for Figure 5, worst case performance on 50 variable problems in terms of compatibility checks

p2	bmcbj+mwo	fccbj+bz	REBA1.0
35	815674	1005	3044
36	2639	1274	2342
37	8067955	1320	2886
38	50716	1828	3092
39	13907031	2103	4787
40	913249	1139	4601
41	2E+08	2737	3997
42	14676577	29868	37618
43	698687	3071	27071
44	1.43E+08	1242863	32790
45	1.57E+08	16877	30992
46	1.56E+08	66307	53962
47	9738619	1875137	48700
48	12706257	107156	52169
49	23113988	524932	50650
50	11733913	269627	100697

Table 5 Data for Figure 6, median performance on 50 variable problems in terms of cpu time

p2	bmcbj+mwo	fccbj+bz	mac+mdo	REBA1.0
35	0	49	250	0
36	0	33	249	0
37	0	49	233	0
38	0	33	233	0
39	0	49	233	0
40	0	49	233	0
41	0	49	233	0
42	0	49	233	0
43	16	49	233	0
44	16	49	216	16
45	16	49	216	16
46	16	49	216	16
47	16	49	216	16
48	16	50	216	33
49	33	66	216	49
50	50	66	216	65

Table 6 Data for Figure 7, worst case performance on 50 variable problems in terms of cpu time

p2	bmcbj+mwo	fccbj+bz	mac+mdo	REBA1.0
35	8016	83	316	49
36	16	83	283	66
37	71233	83	283	66
38	416	82	266	82
39	157783	83	333	82
40	9283	83	266	99
41	1800000	116	850	82
42	138433	916	283	498
43	6916	132	300	283
44	1800000	50266	316	448
45	1800000	583	300	332
46	1800000	2582	950	1082
47	107600	55149	366	766
48	144550	3549	433	815
49	237282	17632	650	732
50	135583	8549	1983	2615

A.2 Results for 100 Variables

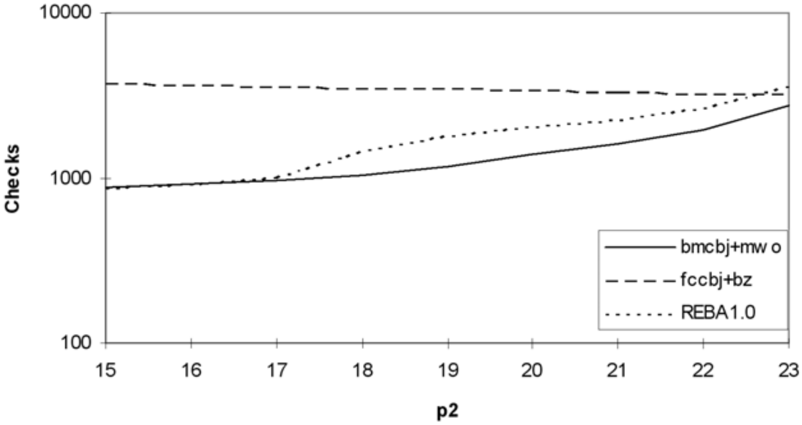


Fig. 14 Median performance on 100 variable problems in terms of compatibility checks

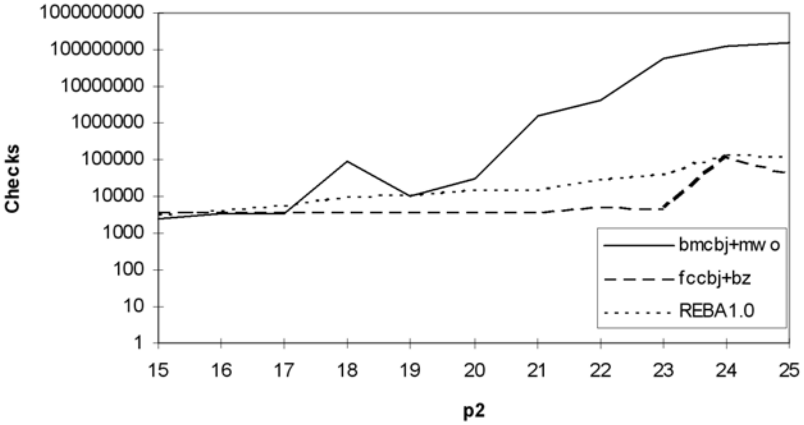


Fig. 15 Worst case performance on 100 variable problems in terms of compatibility checks

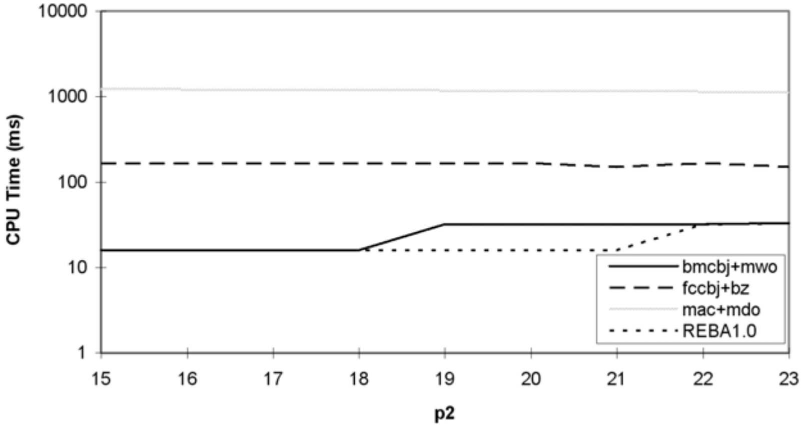


Fig. 16 Median performance on 100 variable problems in terms of cpu time

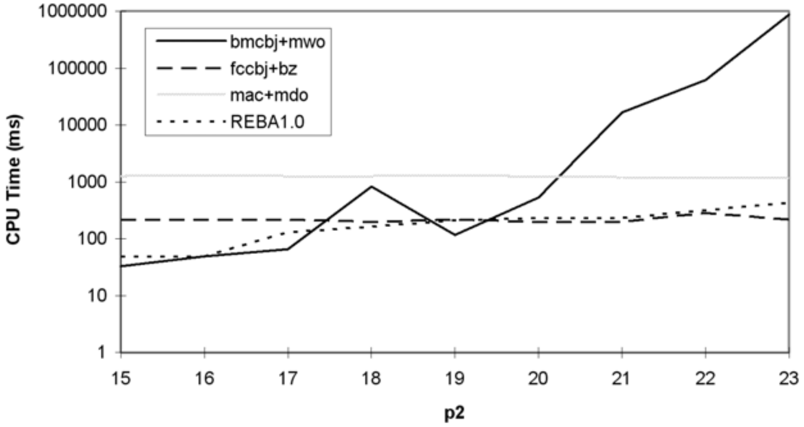


Fig. 17 Worst case performance on 100 variable problems in terms of cpu time

A.3 Tables of results for Figures 14 – 17

Table 7 Data for Figure 14, median performance on 100 variable problems in terms of compatibility checks

p2	bmcbj+mwo	fccbj+bz	REBA1.0
15	865	3757	866
16	918	3682	928
17	971	3607	1010.5
18	1047	3533	1446.5
19	1177.5	3465	1830
20	1383.5	3396	2037.5
21	1598.5	3335	2275.5
22	1940.5	3271	2694
23	2756	3222.5	3606

Table 8 Data for Figure 15, worst case performance on 100 variable problems in terms of compatibility checks

p2	bmcbj+mwo	fccbj+bz	REBA1.0
15	2353	3920	3261
16	3387	3834	4100
17	3527	3773	5996
18	90854	3706	10388
19	10129	3623	11008
20	30716	3584	16016
21	1527678	3804	16256
22	4266115	5348	29393
23	59600500	4400	41197

Table 9 Data for Figure 16, median performance on 100 variable problems in terms of cpu time

p2	bmcbj+mwo	fccbj+bz	mac+mdo	REBA1.0
15	16	166	1232	0
16	16	166	1216	0
17	16	166	1200	16
18	16	166	1199	16
19	32	166	1183	16
20	32	166	1166	16
21	32	150	1166	16
22	32	166	1150	32
23	33	150	1133	33

Table 10 Data for Figure 17, worst case performance on 100 variable problems in terms of cpu time

p2	bmcbj+mwo	fccbj+bz	mac+mdo	REBA1.0
15	33	216	1266	49
16	49	216	1283	49
17	66	216	1266	132
18	832	200	1266	164
19	116	216	1333	215
20	532	200	1249	232
21	16800	200	1216	232
22	61533	283	1199	315
23	884032	216	1182	432

A.4 REBA Results for Different Base Thresholds

Table 11 REBA results for base thresholds of 1.5 and 2.0 for CSPs used in Figures 3–6

p2	median checks		median cpu time		worst case checks		worst case cpu time	
	REBA2.0	REBA1.5	REBA2.0	REBA1.5	REBA2.0	REBA1.5	REBA2.0	REBA1.5
35	299	299	0	0	2068	2057	33	33
36	314	314	0	0	4016	3111	98	49
37	339	339.5	0	0	3421	3143	48	49
38	375.5	380.5	0	0	6000	4709	66	49
39	489.5	520.5	0	0	6356	5837	99	50
40	608.5	602	0	0	6387	6387	82	98
41	716	698	0	0	6567	6028	83	98
42	782.5	762	0	0	8938	8437	98	82
43	895.5	865	0	0	14045	13285	198	116
44	969.5	944	16	0	24296	17351	298	183
45	1181.5	1158	16	16	27050	16279	316	199
46	1461	1408.5	16	16	49321	34802	832	750
47	1730	1744.5	16	16	101768	98090	2482	2533
48	2834	2766	32	32	56755	45932	1098	848
49	4407	4026.5	49	49	133510	146291	2799	3482
50	5962	5172.5	66	66	240225	221734	5682	5132

Table 12 REBA results for base thresholds of 1.5 and 2.0 for CSPs used in Figures 14–17

p2	median checks		median cpu time		worst case checks		worst case cpu time	
	REBA2.0	REBA1.5	REBA2.0	REBA1.5	REBA2.0	REBA1.5	REBA2.0	REBA1.5
15	866	866	0	0	2841	2829	49	49
16	926.5	927	16	0	4197	4191	66	66
17	994.5	997	16	16	4082	4077	50	49
18	1420	1437.5	16	16	13668	10756	115	99
19	1861	1834.5	16	16	7783	7770	66	83
20	2103	2072	16	16	19634	14960	148	216
21	2345	2302.5	16	16	29137	24651	332	299
22	2703.5	2693	32	32	106636	39053	949	366
23	3457	3447	33	33	92259	80427	915	732

References

1. Allen, J.A., Minton, S.: Selecting the right heuristic algorithm: runtime performance predictors. In: Proceedings of 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence, pp. 41–53 (1996)
2. Beck, J.C.: Solution-guided multi-point constructive search for job shop scheduling. Journal of Artificial Intelligence Research 29, 49–77 (2007)
3. Borrett, J.E., Tsang, E.P.K.: A Context for Constraint Satisfaction Problem Formulation Selection. Constraints 6(4), 299–327 (2001)
4. Brélaz, D.: New methods to color the vertices of graphs. Communications of the ACM 22(4), 251–256 (1979)
5. Epstein, S.L., Freuder, E.C., Wallace, R., Morozov, A., Samuels, B.: The adaptive constraint engine. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 525–540. Springer, Heidelberg (2002)
6. Freuder, E.C.: A sufficient Condition for Backtrack-Free Search. Journal of ACM 29, 24–32 (1982)
7. Gagliolo, M., Schmidhuber, J.: Learning restart strategies. In: Veloso, M. (ed.) Proceedings, Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007), Hyderabad, India, January 6-12, pp. 792–797 (2007)
8. Gagliolo, M., Schmidhuber, J.: Impact of censored sampling on the performance of restart strategies. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 167–181. Springer, Heidelberg (2006)
9. Gashnig, J.: A General Backtrack Algorithm That Eliminates Most Redundant Tests. In: Proceedings 5th International Joint Conference on Artificial Intelligence, vol. 457 (1977)
10. Gebruers, C., Hnich, B., Bridge, D., Freuder, E.: Using CBR to select solution strategies in constraint programming. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS, vol. 3620, pp. 222–236. Springer, Heidelberg (2005)
11. Gomes, C.P., Selman, B.: Algorithm portfolios. Artificial Intelligence 126(1-2), 43–62 (2001)
12. Gomes, C., Fernandez, C., Selman, B., Bessiere, C.: Statistical regimes across constrainedness regions. Constraints 10(4), 317–337 (2005)
13. Haralick, R.M., Elliott, G.L.: Increasing Tree Search Efficiency for Constraint Satisfaction Problems. Artificial Intelligence 14, 263–313 (1980)
14. Hoos, H., Tsang, E.P.K.: Local search for constraint satisfaction. In: Rossi, F., van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming, ch. 5, pp. 245–277. Elsevier, Amsterdam (2006)
15. Huberman, B., Lukose, R., Hogg, T.: An economics approach to hard computational problems. Science 265, 51–54 (1997)
16. Kern, M.: Parameter Adaptation in heuristic search - a population-based approach, PhD Thesis, Department of Computer Science, University of Essex, Colchester, UK (2005)
17. Kwan, A.: A framework for mapping constraint satisfaction problems to solution methods, PhD Thesis, Department of Computer Science, University of Essex, Colchester, UK (1997)
18. Mills, P., Tsang, E.P.K., Ford, J.: Applying an Extended Guided Local Search on the Quadratic Assignment Problem. In: Annals of Operations Research, vol. 118, pp. 121–135. Kluwer Academic Publishers, Dordrecht (2003)
19. Nudel, B.: Consistent-Labeling Problems and their Algorithms: Expected-Complexities and Theory-Based Heuristics. Artificial Intelligence 21, 135–178 (1983)
20. Prosser, P.: Hybrid Algorithms for the Constraint Satisfaction Problem. Computational Intelligence 9, 268–299 (1993)

21. Prosser, P.: Binary Constraint Satisfaction Problems: Some are Harder than Others. In: Proceedings 11th European Conference on Artificial Intelligence, pp. 95–99 (1994)
22. Puget, J.-F.: Applications of constraint programming. In: Montanari, U., Rossi, F. (eds.) Proceedings, Principles and Practice of Constraint Programming (CP 1995). LNCS, pp. 647–650. Springer, Heidelberg (1995)
23. Rossi, F., van Beek, P., Walsh, T. (eds.): Handbook of Constraint Programming. Elsevier, Amsterdam (2006)
24. Sabin, D., Freuder, E.C.: Contradicting Conventional Wisdom in Constraint Satisfaction. In: Proceedings 11th European Conference on Artificial Intelligence, pp. 125–129 (1994)
25. Smith, B., Grant, A.: Sparse Constraint Graphs and Exceptionally Hard Problems. In: Proceedings 14th International Joint Conference on Artificial Intelligence, pp. 646–651 (1995a)
26. Smith, B., Grant, A.: Where the *Exceptionally* Hard Problems are. In: Workshop on Studying and Solving Really Hard Problems. CP 1995, pp. 172–182 (1995b)
27. Smith, B.: Phase Transition and the Mushy Region in Constraint Satisfaction Problems. In: Proceedings 11th European Conference on Artificial Intelligence, pp. 100–104 (1994a)
28. Smith, B.: In search of Exceptionally Difficult Constraint Satisfaction Problems. In: Proceedings of the Workshop on Constraint Processing, 11th European Conference on Artificial Intelligence, pp. 79–86 (1994b)
29. Turner, J.S.: Almost all k-Colorable Graphs are Easy to Color. Journal of Algorithms 9, 63–82 (1988)
30. Tsang, E.P.K., Borrett, J.E., Kwan, A.C.M.: An Attempt to Map a Range of Constraint Satisfaction Algorithms and Heuristics. In: Proceedings AISB 1995, pp. 203–216 (1995)
31. Tsang, E.P.K.: Foundations of Constraint Satisfaction. Academic Press, London (1993)
32. Voudouris, C., Tsang, E.P.K.: Guided local search. In: Glover, F. (ed.) Handbook of metaheuristics, pp. 185–218. Kluwer, Dordrecht (2003)
33. Voudouris, C., Dorne, R., Lesaint, D., Liret, A.: iOpt: A Software Toolkit for Heuristic Search Methods. In: Walsh, T. (ed.) CP 2001. LNCS, vol. 2239, pp. 716–729. Springer, Heidelberg (2001)
34. Voudouris, C., Owusu, G., Dorne, R., Lesaint, D. (eds.): Service Chain Management: Technology Innovation for the Service Business. Springer, Heidelberg (2008)
35. Wolpert, D.H., Macready, W.G.: No Free Lunch Theorems for search, Technical Report SFI-TR-95-02-010, Santa Fe Institute (1995)
36. Wolpert, D.H., Macready, W.G.: No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
37. Minton, S.: Automatically configuring constraint satisfaction programs, a case study. Constraints 1(1-2), 7–43 (1996)
38. Kautz, H., Horvitz, E., Ruan, Y., Gomes, C., Selman, B.: Dynamic restart policies. In: Proceedings, Eighteenth National Conference on Artificial Intelligence (AAAI 2002), Edmonton, Alberta, Canada, pp. 674–682 (2002)
39. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: SATzilla: portfolio-based algorithm selection for SAT. Journal of Artificial Intelligence Research 32, 565–606 (2008)

Part IV
Multi-Agent Systems

Collaborative Computational Intelligence in Economics

Shu-Heng Chen

Abstract. In this chapter, we review the use of the idea of collaborative computational intelligence in economics. We examine two kinds of collaboration: first, the collaboration within the realm of computational intelligence, and, second, the collaboration beyond the realm of it. These two forms of collaboration have had a significant impact upon the current state of economics. First, they enhance and enrich the heterogeneous-agent research paradigm in economics, alternatively known as agent-based economics. Second, they help integrate the use of human agents and software agents in various forms, which in turn has tied together agent-based economics and experimental economics. The marriage of the two points out the future of economic research. Third, various hybridizations of the CI tools facilitate the development of more comprehensive treatments of the economic and financial uncertainties in terms of both their quantitative and qualitative aspects.

1 Introduction

Computational intelligence has been applied to economics for more than a decade. These applications can be roughly divided into two categories, namely, *agent-based computational economics* and *financial data mining*. Although in many such studies only one computational intelligence (CI) tool is involved, studies which apply more than one CI tool also exist and have become popular. In these studies, a few CI tools *work together* or *collaborate* with each other to perform a certain function. These studies are, therefore, examples of the use of *collaborative computational intelligence.* In this chapter, we shall provide a general review of collaborative computational intelligence in economics based on these studies.

There are three major sources that motivate the application of collaborative computational intelligence (CCI) to economics. The first source of stimulation

Shu-Heng Chen
Department of Economics, National Chengchi University
e-mail: chchen@nccu.edu.tw

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 233–273.
springerlink.com

comes from the new research paradigm in macroeconomics, which is known as the *heterogeneous-agent research paradigm*. This new research paradigm is an alternative to the conventional *representative-agent paradigm*, which has dominated the development of macroeconomics for about half a century [54]. In Section 2, we shall comprehensively review this development and point out its relevance to CCI.

The second source pertains to the recent attempt to integrate the *experimental economics* and *agent-based computational economics* [22, 27, 45, 85]. One main task associated with this integration is an inquiry into the relationship between human agents and software agents. Three relations have been mentioned in the literature, namely, *mirroring*, *competition*, and *collaboration*. In Section 3, we shall review this development and indicate how CCI can be applied along these lines.

The last source is the use of hybrid systems in *financial data mining* [16, 109]. Over the last two decades, we have evidenced the simultaneous use of multiple CI tools to build intelligent systems, including many financial ones. However, a general review of this development is beyond the scope of this chapter. Besides, related discussions may be available from other chapters. Therefore, to avoid redundancy and to use the limited size efficiently, in Section 4, we shall focus on only the few frequently used economic and financial hybrid systems in financial data mining. Section 5 will give the concluding remarks.

2 Heterogeneous Agents

Why is collaborative computational intelligence relevant to the study of economics? There is a straightforward answer: economic agents are *heterogeneous*, and their differences and interactions match the idea of CCI well. In this section, we shall see how computational intelligence has worked with the conventional approach in modeling a population of heterogeneous agents and their interactions. Basically, these types of collaboration can be differentiated into three levels, from the *macroscopic*, to the *microscopic*, to the *molecule* level. We shall first briefly state these three levels of collaboration (Section 2.1), and then elaborate on the significance of the collaboration at each level by highlighting existing research (Sections 2.2-2.4).

2.1 The Three Levels of Collaboration

From a *macroscopic* viewpoint, the population-based CI tools, such as *evolutionary computation* and *swarm intelligence*, have already encapsulated the idea of *population dynamics*. Therefore, they are readily applicable to model a population of heterogeneous economic agents. As we shall see in Section 2.2, there has already been a great deal of this kind of collaboration. On the other hand, from a *microscopic* viewpoint, we may also represent the heterogeneity of agents *individually* by different CI algorithms. In this case, these agents can be regarded as incarnations of different CI algorithms. These algorithms may be the same and may only differ in their control parameters, or else they may be fundamentally different. For example, in the former case, all agents are represented by genetic programming, while using different parameters of population size [36], or, for the latter case, some agents are represented by the *K-nearest-neighbors* (KNN) algorithm or general *instance-based learning* (IBL) algorithms, while some others are represented by Bayesian learning [21].

By moving further down to even more fine-detail level, referred to as the *molecule level*, we can regard each individual agent as being represented by more than one CI tool [67, 98]. In other words, the idea of *hybrid systems* is applied to model agents, and hence their heterogeneity can also be manifested in terms of different hybridization styles.

2.2 Macroscopic Level: Evolving Population

From the macroscopic viewpoint, the collaboration of computational intelligence with the conventional *cluster analysis* has developed the conventional economic agent engineering from the *N-type designs* into the *autonomous-agent designs*. This development is particularly evident in the *agent-based financial modeling*. To review this progress, let us first briefly review how agent-based economic and financial modeling have arisen (Section 2.2.1), and then consider the two different approaches in the designs of economic and financial agents (agent engineering) (Sections 2.2.2-2.2.3).

2.2.1 Agent-Based Economic Modeling

The rise of agent-based economics and finance can be considered to be a paradigm shift after long questioning, and even dissatisfaction with, the mainstream economics methodology built upon *representative agents*. [54] provides a lengthy discussion on this "troubling" concept. There are many reasons for going against the device of the representative agents, both from the empirical and theoretical aspects. The main empirical grounds are that there is ample empirical evidence to show that great heterogeneity and diversity exists at the micro level, from households, firms, traders, and other decision-makers. Nonetheless, a solid understanding of this diversity, such as the wealth distribution of households, the size distribution of firms, and the optimistic and pessimistic forecasting distributed among financial practitioners, is still lacking. Therefore, there is a need to search for a more suitable methodology in order to study the distributive behavior of an economy. Furthermore, given the great diversity at the micro level, the relationship between the macro (aggregates) and micro (individuals) becomes much more complex than that of which holds in the representative-agent economy. The exact relationship between the micro and macro has actually presented economic theorists with a new challenge. All these together motivated the formation of heterogeneous-agent approaches or the agent-based paradigm approach to economics in the 1990s.[1]

[1] There are a number of textbooks on macroeconomics written within this new background. See, for example, [5, 44].

Agent-based economics is an application of agent-based tools to economics. As with other agent-based models, the agent-based economics also begin with *agents*. This starting procedure is mainly composed of technical characterizations of economic agents, i.e., how to design economic agents. Hence, it is also called *economic agent engineering*. Economic agent engineering matters because, in general, it is expected that different designs of agents can result in different aggregate dynamics, even under the same institutional arrangements. Therefore, the agent-based economic model can serve as a tool to run a sensitivity analysis of a specific market or institution design to evaluate its possible performance. In this sense, it enables social scientists to have their own laboratory and to perform their own experiments as natural scientists do.

What follows are two very different designs of financial agents. The first one is called the *N-type design* (Section 2.2.2), whereas the second is called the *autonomous-agent design* (Section 2.2.3).

2.2.2 N-Type Designs

Let us now focus on the core of the agent-based financial markets, namely, financial agents and their design. In reality, financial agents can differ in many dimensions, ranging from expectations formation (beliefs), trading strategies, information exposure, risk attitudes,and wealth (investment scale), to the need for liquidity, etc. Given this high-dimensional heterogeneity, the essential question for financial agent engineering is to decide how much heterogeneity is to be reflected in the artificial markets. How much coarsely or finely do we want to differentiate these financial agents?

Before we examine the design of artificial financial agents, it is useful to recall what we have done for other artifacts. To name a few, the design of artificial ants (*ant algorithms*) was motivated by observing the behavior of real ants in a laboratory; the design of artificial bacteria (*bacterial algorithms*) was inspired by the microbial evolution phenomenon; the design of the artificial brain (*neural networks*, *self-organizing maps*) was motivated by the study of the real human brain; and the design of the evolutionary process (*evolutionary algorithms*) was inspired by real biological evolution. Generally speaking, the design of an artifact is, by and large, motivated and guided by the behavior of its counterpart in nature.

The design of artificial financial agents is no exception. It is highly motivated by observing how real financial agents behave. Empirical evidence accumulated since the late 1980s and early 1990s has shed new light on the forecasting behavior of financial agents. This empirical evidence was obtained through different kinds of surveys, such as questionnaires and telephone interviews, with financial specialists, bankers, currency traders, and dealers, etc. [50, 2]. The general findings from these abundantly established empirical data are two-fold. First, the data indicate that, by and large, there are two kinds of expectations existing in the market. The one which is characterized as a stabilizing force of the market is associated with a type of financial agent, called the *fundamentalist*. The one which is characterized as a destabilizing force is associated with another type of financial agent, called the *chartist*, *technical analyst* or *trend extrapolator*. Second, the proportion (micro-structure) of fundamentalists and chartists, also called the *market fraction*, is changing over time, which indicates that financial agents are adaptive. These empirical findings provide the initial direction for the early development of financial agent engineering. First, they suggest what rules to look at; second, they point out the significance of learning and adaptation.

Fundamentalists and chartists are concerned with two very different beliefs regarding the stock price dynamics. In a simple setting, they differ in terms of the mean-reverting speed of the stock price when it is mispriced (undervalued or overvalued). Fundamentalists tend to believe that the mispriced situation will soon be corrected, whereas chartists tend to believe that in the short run it will continue.

2-Type Design

To make what we say more precise, we generally denote the forecasting rule of a type-h agent as follows:

$$E_{h,t}[p_{t+1}] = f_{h,t}(p_t, p_{t-1}, \ldots), \tag{1}$$

where $E_{h,t}$ refers to the expectations of the type-h agent at time t. Equation (1) indicates the one-step ahead forecast. At the beginning, we start with a very general forecasting function $f_{h,t}$, which uses all the historical data on price up to the present. In addition, by considering that agents are adaptive, we allow the function to change over time and hence denote it by the subscript t.

For the fundamentalists ($h = f$) and chartists ($h = c$), their forecast rules, in a very simple setting, can be written as

$$E_{f,t}[p_{t+1}] = p_t + \alpha_f(p_t^f - p_t), \;\; 0 \leq \alpha_f \leq 1., \tag{2}$$

$$E_{c,t}(p_{t+1}) = p_t + \alpha_c(p_t - p_{t-1}), \;\; 0 \leq \alpha_c. \tag{3}$$

The idea of these two behavioral rules is that the fundamentalist has a *mean-reverting* belief, and his belief is characterized by a reverting coefficient (α_f), whereas the chartist has a trend-continuing belief, and his belief is characterized by an extrapolating coefficient (α_c). The magnitude of the reverting coefficient (α_f) measures the speed at which the fundamentalists expect the price to return to the fundamental one (p_t^f), whereas the magnitude of the extrapolating coefficient (α_c) expresses the degree to which chartists expect the past to change into the future.

3-Type Design

There is little doubt that the behavior of financial agents can be more complex than the two-type design. One obvious way to scale-up this design is to add more types of agents to the model so as to take into account a finer degree of heterogeneity of financial agents. This type of expansion is called the *N-type design*. For example, in a three-type design, one can further distinguish two kinds of chartists, namely, *momentum traders* and *contrarian traders*, or simply, *contrarians*. Like momentum traders, contrarians extrapolate past movements of the price into the future, but they follow the opposite of the trend. More precisely, their forecasting rule is as follows:

$$E_{co,t}(p_{t+1}) = p_t + \alpha_{co}(p_t - p_{t-1}), \ \ \alpha_{co} \leq 0. \tag{4}$$

Contrarians consider that the price trend will finish soon, and will start to reverse. However, unlike fundamentalists, contrarians do not base their forecasts on the fundamental price, which they either do not know, or they do not care about.

The recent availability of more proprietary data has enhanced the transparency of the trading behavior of financial agents, including both individual and institutional investors. Empirical studies using such data have shown that individuals and institutions differ systematically in their reaction to past price performance and the degree to which they follow momentum and contrarian strategies. On average, individual investors are contrarian investors: they tend to buy stocks that have recently underperformed the market and sell stocks that have performed well in recent weeks [15]. With this empirical basis, financial agent engineering has already added the contrarians to the fundamentalist-chartist model, and popularized this three-type design.

Generalization of 2- and 3-Type Designs

Financial agent engineering can also be advanced by enriching the behavioral rules associated with each type of financial agent. This alteration may make financial agents more interdisciplinary. Considerations from different fields, including neural sciences, cognitive psychology, and statistics, can be incorporated into designs. For example, in behavioral finance, there is a psychological bias known as the "*law of small numbers*", which basically says that people underweight long-term averages, and tend to put too much weight on recent experiences (the recency effect). When equity returns have been high for many years, financial agents with this bias may believe that high equity returns are "normal". By design, we can take such bias into account. One way to do so is to add a *memory parameter* to the behavioral rules of our financial agents. This more general rule for contrarians is specified as follows:

$$E_{c,t}(p_{t+1}) = p_t + \alpha_c(1-\beta_c)\sum_{i=0}^{T}(\beta_c)^i(p_{t-i} - p_{t-i-1}), \ \ 0 \leq \alpha_c, \ \ 0 \leq \beta_c \leq 1. \tag{5}$$

$$E_{co,t}(p_{t+1}) = p_t + \alpha_{co}(1-\beta_{co})\sum_{i=0}^{T}(\beta_{co})^i(p_{t-i} - p_{t-i-1}), \ \ 0 \geq \alpha_{co}, \ \ 0 \leq \beta_{co} \leq 1. \tag{6}$$

The momentum traders and contrarians now compute a moving average of the past changes in the stock price and they extrapolate these changes into the future of the stock price. However, we assume that there is an exponential decay in the weights given to the past changes in the stock price. The parameters β_c and β_{co} can be interpreted as reflecting the memory of momentum traders and contrarians. If $\beta_c = \beta_{co} = 0$, momentum traders and contrarians remember only the last period's price change and they extrapolate this into the future. When β_c and β_{co} increase, the weight given to the price changes farther away in the past increases. In other words, the chartists' memory becomes longer.

The psychological bias mentioned earlier, therefore, corresponds to a small value of this memory parameter, and this "hypothesis" can actually be tested. In fact, by using the data for the S&P 500 index, one of the three major US stock market indices, from January 1980 to December 2000, [4] actually estimated a three-type agent-based financial market model, and found that contrarians have a longer memory than momentum traders when they form their forecast of the future price. Of course, this is just the beginning in terms of seeing how agent-based financial market models can be quantified so as to communicate with behavioral finance.

Adaptive Behavior

In the original fundamentalist-chartist model, learning does not exist. Agents who initially happen to be fundamentalists will continue to be fundamentalists and will never change this role, and likewise for chartists. As a result, the proportion (market fraction) of fundamentalists and chartists remains fixed. Nonetheless, this simplification underestimates the uncertainty faced by each trader. In general, traders, be they fundamentalists or chartists, can never be certain about the duration of the biased trend, since the trend can finish in weeks, months, or years. This uncertainty causes the alerted traders to review and revise their beliefs constantly. In other words, traders are *adaptive*.

Therefore, a further development of financial agent engineering is to consider an evolving micro-structure of market participants. In this extension, the idea of adaptive agents or learning agents is introduced into the model. Hence, an agent who was a fundamentalist (chartist) may now switch to being a chartist (fundamentalist) if he considers this switching to be more promising. Since, in the two-type model, agents can only choose to be either a fundamentalist or a chartist, modeling their learning behavior becomes quite simple, and is typically done using a *binary-choice model*, specifically, the *logit model* or the *Gibbs-Boltzmann distribution*.

The logit model, also known as the *Luce model*, is the main model used in the psychological theory of choice, and was proposed by Ducan Luce in 1959 in his seminal book, "*Individual Choice Behavior: A Theoretical Analysis*." Consider two alternatives f (fundamentalist) and c (chartist). Each will produce some gains to the agent. However, since the gain is random, the choice made by the agent is random as well. The logit model assumes that the probability of the agent choosing f is the probability that the profits or utilities gained from choosing f are greater than those gained from choosing c. Under a certain assumption for the random component of the utility, one can derive the following *binary logit model*:[2]

$$Prob(X = f,t) = \frac{\exp^{\lambda V_{f,t-1}}}{\exp^{\lambda V_{f,t-1}} + \exp^{\lambda V_{c,t-1}}}, \tag{7}$$

[2] The extension into the multinomial logit model is straightforward.

where $V_{f,t}$ and $V_{c,t}$ are the deterministic components of the gains from the alternatives f and c at time t. The parameter λ is a parameter carried over from the assumed random component. The logit model says that the probability of choosing the alternative f depends on its *absolute deterministic advantages*, as we can see from the following reformulation:

$$Prob(X = f,t) = \frac{1}{1 + \exp^{-(\lambda(V_{f,t-1} - V_{c,t-1})}}. \tag{8}$$

When applied to the agent-based financial models, these deterministic components are usually related to the temporal realized profits associated with different forecasting rules. So, in the two-type model, if V_f can be the temporal realized profits from being a fundamentalist, then V_c can be the temporal realized profits from being a chartist. In addition, there is a new interpretation for the parameter β, namely, the *intensity of choice*, because it basically measures the extent to which agents are sensitive to the additional profits gained from choosing f instead of c.

Market Maker Equation

The market fractions above then determine the market fraction of each type of agent in the market. For example, if $Prob(X = F) = 0.8$, it means that 80% of the market participants are fundamentalists and the remaining 20% are chartists. The asset price will be determined by this market fraction via the *market maker equation*.

$$p_t = p_{t-1} + \mu_0 + \mu_1 D_t \tag{9}$$

where

$$D_t = \sum_h w_{h,t} d_{h,t} = \sum_h Prob(X = h,t) d_{h,t}. \tag{10}$$

Equation (9) is the market maker equation, which assumes that the price is adjusted by the *market maker*, whose decision is in turn determined by the excess demand normalized by the number of market participants, D_t. D_t, in Equation (10), is a weighted average of the individual demand of each type of trader, weighted by the market fractions (7).

Risk Preference and Portfolio

The demand for assets of each type of trader is derived in a standard expected-utility maximization manner, which depends on the *risk preference* of the type-*h* agent. Risk preference is important because it is the main determinant of agents' portfolios, i.e., how agents' wealth is distributed among different assets. The classical *Markowitz mean-variance portfolio selection model* offered the first systematic treatment of asset allocation. Harry Markowitz, who later received the 1990 Nobel Prize in Economics for this contribution, assumes that investors are concerned only with the mean and variance of returns. This *mean-variance preference* has been extensively applied to modeling agents' risk preference since the variance of returns is normally accepted as a measure of risk.

In addition to the mean-variance preference, there are two other classes of risk preferences that are widely accepted in the standard theory of finance. These two correspond to two different attitudes toward risk aversion. One is called *constant absolute risk aversion* (CARA), and the other is called *constant relative risk aversion* (CRRA). When an agent's preference exhibits CARA, his demand for the risky asset (or stock) is independent of his changes in wealth. When an agent's preference exhibits CRRA, his demand for risky assets will increase with wealth in a linear way. Using a Taylor expansion, one can connect the mean-variance preference to CARA preferences and CRRA preferences. In fact, when the returns on the risky assets follow a normal distribution, the demand for risky assets under the mean-variance preference is the same of that under the CARA preference, and is determined by the *subjective-risk-adjusted expected return*.

$$d_{h,t} = \frac{E_{h,t}(\Delta p_{t+1})}{a_{h,t}V_{h,t}(\Delta p_{t+1})} = \frac{E_{h,t}(p_{t+1}) - p_t}{a_{h,t}V_{h,t}(\Delta p_{t+1})}, \tag{11}$$

where $\Delta p_{t+1} = p_{t+1} - p_t$, and $a_{h,t}$ is a risk aversion coefficient. The $E_{h,t}(p_{t+1})$ in the numerator of Equation (11) is given by Equations (2), (3) and (4), and $V_{h,t}$ in the denominator represents the perceived risk of the type-h agents. Further details of the formation of this subjective perceived risk can be found in the agent-based finance literature [42, 58].

Use of the N-Type Designs

While putting this N-type design into practical financial forecasting is still in its infancy stage, we have already seen some successful initial attempts in foreign exchange markets, which can be found in [43], a three-type design, and [75], a two-type design.

2.2.3 Autonomous-Agent Designs

So far, all the types and rules of financial agents are given at the beginning of the design, and what financial agents can do is to choose among these different types and rules based on their past experiences. The N-type design has characterized a major class of agent-based financial markets. However, this way of doing things also severely restricts the degree of autonomy available for financial agents. First, they can only choose how to behave based on what has been offered; secondly, as a consequence, there will be no new rules available unless they are added outside by the designers. If we want our artificial financial agents to behave more like real financial agents, then we will certainly expect that they learn and discover *on their own*. Therefore, as time goes by, new rules which have never been used before and have not been supplied by the designer may be discovered by these artificial agents inside the artificial world.

Genetic Algorithms

Designing artificial agents who are able to design on their own is an idea similar to John von Neumann's *self reproducing automata*, i.e., a machine which can reproduce itself. This theory had a deep impact on John Holland, the father of the *genetic algorithm*indexgenetic algorithms. Under von Neumann's influence, Holland had devoted himself to the study of a general-purpose computational device that could serve as the basis for a general theory of automata. In the 1970s, he introduced the genetic algorithm, which was intended to replace those ad hoc learning modules in contemporary mainstream AI. Using genetic algorithms, Holland could make an adaptive agent that not only learned from experience but could also be spontaneous and creative. The latter property is crucial for the design of artificial financial agents. In 1991, Holland and John Miller, an economist, published a sketch of the artificial adaptive agent in the highly influential American Economic Review. This blueprint was actually carried out in an artificial stock project in 1988 in the Santa Fe Institute [82, 10].

Sante Fe Institute Artificial Stock Market

Armed with GAs, the *Santa Fe Artificial Stock Market* (SFI-ASM) considers a novel design for financial agents. First, like many N-type designs, it mainly focuses on the forecasting behavior of financial agents. Their trading behavior, as depicted in Equation (11), will depend on their forecasts of the price in the next period. Second, however, unlike the N-type designs, these agents are not divided into a fixed number of different types. Instead, the forecasting behavior of each agent is "customized" via a GA. We shall be more specific regarding its design because it provides us with a good opportunity to see how economists take advantage of the increasing computational power to endow artificial decision makers with a larger and larger degree of autonomy.

In the SFI-ASM, each financial agent h uses a linear forecasting rule as follows:

$$E_{h,t}(p_{t+1}) = \alpha_{h,t} + \beta_{h,t} p_t. \tag{12}$$

However, the coefficients $\alpha_{h,t}$ and $\beta_{h,t}$ not only change over time (time-dependent), but also are state-dependent. That is, the value of these two coefficients at time t will depend on the state of the economy (market) at time t. For example, the recent price dynamics can be an indicator, so, say, if the price has risen in the last 3 periods, the financial agent may consider lower values of both α and β than otherwise. The price dividend ratio can be another indicator. If the price dividend ratio is lower than 50%, then the financial agent may want to take a higher value of β than if it is not. This state-dependent idea is very similar to what is known as *classification and regression trees* (CART) or *decision trees*, a very dominant approach in machine learning.

Therefore, one simple way to think of the artificial agents in the SFI-ASM is that they each behave as machine-learning people who use *regression trees* to forecast the stock price. At each point in time, the agent has a set of indicators which help him to decompose the state of the economy into m distinct classes, $(A^1_{h,t}, A^2_{h,t}, \ldots, A^m_{h,t})$, and corresponding to each of the classes there is an associated linear forecasting model. Which model will be activated depends on the state of the market at time t, denoted by S_t. Altogether, the behavior of the financial agent can be summarized as follows:

$$E_{h,t}(p_{t+1}) = \begin{cases} \alpha^1_{h,t} + \beta^1_{h,t} p_t, & if \;\; S_t \in A^1_{h,t}, \\ \alpha^2_{h,t} + \beta^2_{h,t} p_t, & if \;\; S_t \in A^2_{h,t}, \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \alpha^m_{h,t} + \beta^m_{h,t} p_t, & if \;\; S_t \in A^m_{h,t}. \end{cases} \tag{13}$$

A few remarks are added here. First, the forecasting rule summarized above is updated as time goes by, as we keep the subscript t there. So, agents, in this system, are learning over time with a regression tree, or they are using a time-variant regression tree, in which all the regression coefficients and classes may change accordingly with the agents' learning. Second, agents are completely heterogeneous as we also keep the subscript h above. Therefore, if there are N financial agents in the markets at each point in time, we may observe N regression trees, each of which is owned and maintained by one individual agent. Third, however, the forecasting rules introduced in the SFI-ASM are not exactly regression trees. They are, in fact, *classifier systems*.

Classifier System

A classifier system is another of John Holland's inventions in the late 1970s. This system is similar to the Newell-Simon type of expert system, which is a population of if-then or condition-action rules. The conventional expert systems are not able to learn by themselves. To introduce adaptation into the system, Holland applied the idea of market competition to a society of if-then rules. A formal algorithm, known as the *bucket-brigade algorithm*, credits rules generating good outcomes and debits rules generating bad outcomes. This accounting system is further used to resolve conflicts among rules. The shortcoming of the classifier system is that it cannot automatically generate or delete rules. Therefore, a GA is applied to evolve them and to discover new rules.

This autonomous-agent design has been further adopted in many later studies. While most studies continuously carried out this task using genetic algorithms[3], a few studies also used other population-based learning models, such as evolutionary programming and genetic programming.

[3] A lengthy review of this literature can be found in [23].

Genetic Programming and Autonomous Agents

The development from the few-type designs to the many-type designs and further to the autonomous-agent designs can be considered to be part of a continuous effort to increase the collective search space of the forecasting function $E_{h,t}$, from finite to infinite space, and from parametric to semi-parametric functions. The contribution of genetic programming (GP) to this development is to further extend the search space to a infinite space of non-parametric functions, whose *size* (e.g., the dimensionality, the cardinality or the number of variables used) and *shapes* (for example, linearity or non-linearity, continuity or discontinuity) have to be determined, via search, simultaneously. This way of increasing the degree of autonomy may not contribute much to the price dynamics, but can enrich other aggregate dynamics as well as the behavior at the individual level. As we shall see below, the endogenous determination of the size and shape of $E_{h,t}$ provides us with great opportunities to see some aspects of market dynamics which are not easily available in the N-type designs or other autonomous-agent design.

The first example concerns *the sophistication of agents* in market dynamics. The definition and operation of GP rely on a specific language environment, known as LIST Programming (LISP). For each LISP program, there is a tree representation. The number of nodes (leaves) or the number of depths in the LISP trees provides one measure of complexity in the vein of the *program length*. This additional observation enables us to study not just the heterogeneity in $E_{h,t}$, but also the associated complexity of $E_{h,t}$. In other words, genetic programming can not only distinguish agents by their forecasts, as the N-type designs did, but further delineate the differentiation according to the agents' sophistication (complexity). Must the survival agents be sophisticated or can the simple agents prosper as well?

One interesting hypothesis related to the above inquiry is the *monotone hypothesis*: the degree of traders' sophistication is an increasing function of time. In other words, traders will evolve to be more and more sophisticated as time goes on. However, this hypothesis is rejected in [33]. They found that, based on the statistics on the node complexity or the depth complexity, traders can evolve toward a higher degree of sophistication, and at some point in time, they can be simple as well.

The second example concerns the capability to distinguish the information from noise. As we mentioned earlier, the variables recruited in the agents' forecasting function are also endogenously determined. This variable-selection function allows us to examine whether the smart picking of these variables is crucial for survival. In particular, the hypothesis of the extinction of noisy traders says that traders who are unable to distinguish information from noise will become extinct. [34] test this hypothesis. In an agent-based artificial market, they supplied traders with both informative and noisy variables. The former include prices, dividends and trading volumes, whereas the latter are just series of pseudo random numbers. Their simulation shows, as time goes on, that traders who are unable to distinguish information from noise do have a tendency to decline and even become extinct.

2.3 Microscopic Level: Heterogeneity in Intelligence

2.3.1 Bounded Rationality and Intelligence Quotient

At the microscopic level, collaborative computational intelligence has shown its relevance to modeling *bounded rationality*. Computational intelligence can be collaborated to address bounded rationality because different CI tools themselves may already demonstrate different degrees of rationality or intelligence. Having said that, we are aware of the measurement problems pertaining to rationality or intelligence. Certainly, so far, there is no formal measure of rationality, and whether it can be positively related to the *intelligence quotient* (IQ) is also unclear[4], even though the latter is frequently used as a proxy for the former.[5] In addition, our experience that smart people are not immune from doing dumb things further casts doubt on the connection between the two.[6] Needless to say, the study of human intelligence is still an open-ended on-going body of research. The current research trend in empirical economics, however, has attempted to make the behavior of bounded-rational agents transparent or observable using real-world data. In addition, bounded rationality is frequently used as an input in models since it may generate different predictions or outcomes.

Therefore, despite the lack of an acceptable measure of rationality, the current trend in economic research forces us to ask how computational intelligence can help us building economic models of bounded rationality or building bounded-rational agents. The simplest answer is that CI tools can help us to model the *learning* or *adaptive behavior* of bounded-rational agents. A huge economic literature has already documented this development.[7] Almost all major CI tools have been applied to model the learning and adaptive behavior of economic agents, that includes reinforcement learning, instance-based learning, regression trees, Bayesian learning, artificial neural nets, fuzzy logic, and evolutionary computation [23].

However, these studies have been frequently criticized as *ad hocry* in terms of the choice of a specific CI tool. Hence, to move forward, the research question to address is: can we have a theory or an acceptable practice to guide us in the choice of CI tools when modeling the adaptive economic agents? This question has motivated

[4] Despite their incurring criticisms, some empirical studies support a positive correlation between IQ and income. While the correlation coefficient is often found to be less than 0.5, it may increase with age to some extent [57, 61].

[5] The most famous example is the device of the *zero-intelligence agents* introduced in [52]. To motivate this design, [52] raised the issue: *how much intelligence is required of an agent to achieve human-level trading performance?* The zero-intelligence agent, based on the design of [52], is a *randomly behaved agent*, who needs no memory, no learning, and no strategic playing. It is a kind of naive agent, who just randomly bids or randomly asks. It was found that these randomly behaved agents are sufficiently able to replicate the market efficiency achieved by human agents.

[6] That is why we frequently see books like [48].

[7] Among many available textbooks, [89] is the first one which introduces the materials of computational intelligence to economists. [47] also has a section introducing the use of computational intelligence to build learning models.

a number of research directions. The one which is related to *experimental economics* will be addressed in Section 3. In this subsection, we address the one directly related to *controlling the degree of intelligence or smartness*. However, before that, let us make a distinction between the two.

Certainly, models of learning and adaptation should be a part of bounded rationality, but not the whole. What has been generally neglected in the past applications of CI tools to learning is that *economic agents are not equally smart*, as they have different IQ, EQ, or whatever Q.[8] Even though they are all learning, it does not mean that they are learning under the same cognitive constraints or under the same mental capacity. In fact, current *behavioral genetics* enhance our understanding of the heterogeneity in human cognitive ability, that includes the ability to memorize, to search, to learn, to perceive, and to socialize [76, 84]. A large proportion of the variance in cognitive abilities can be attributed to genetics. Therefore, the difference in *genome* may need to be incorporated into our applications of CI tools, and agents with different innate cognitive abilities are expected to be equipped with different CI tools.

2.3.2 Heterogeneity Characterized by Different CI Tools

Fortunately, taxonomies of CI tools based on the degree of cognitive constraints are possible, while not perfect.[9] For example, reinforcement learning models tend to be regarded as less sophisticated than evolutionary computational models. Therefore, a market composed of agents with heterogeneity in intelligence can be considered to be a market composed of some "less smart" agents, whose adaptive behavior is driven by reinforcement learning, and some "smart" agents, whose adaptive behavior is driven by evolutionary computation. With this mixture, a few CI tools can *interact* with each other in the same economic environment. This defines the first kind of application of CCI at the micro level.[10]

[21] is probably the first study of this kind. In a context of agent-based artificial stock markets, [21] considers three different types of agents. The first is the momentum trader (chartist), whose forecasting behavior is a special case of Equation (3) when α_c is set to 1 and becomes [11]

$$E_{c,t}(p_{t+1}) = p_t + (p_t - p_{t-1}). \tag{14}$$

[8] Of course, IQ as an important causal determinant of decision making is not just neglected in economics, but is neglected in all social sciences [69].

[9] Both [17] and [45] give a comprehensive treatment of various learning algorithms.

[10] While various computational intelligence tools are often compared or compete in the financial engineering domains, such as financial forecasting, trading, etc.. There has been little work on comparing their behavior in agent-based economic modeling. The difference between the two study styles is that in the former case the competition or tournament is conducted without interaction or feedback, whereas in the latter case this mechanism is presented.

[11] See Section 2.2.2 for a detailed description of momentum traders.

This type of momentum trader is naive in the sense that they continuously believe that what they experience today regarding the price change will remain unchanged tomorrow. Given this naive momentum trader, they also introduced two sophisticated type of agents, namely, *empirical Bayesian traders* and *K-nearest-neighbor traders*. Both *empirical Bayesian traders* and *K nearest neighbors* (KNN) are active members of the CI toolkit.

Bayesian Learning

Before the advent of computational intelligence in the early 1990s, Bayesian learning was the dominant learning model used by economists. Economists have a strong preference for Bayesian learning partially because in spirit it is consistent with optimization. The optimality of Bayesian learning has been well established in *statistical decision theory*. It has a lot of variants and applications in regard to the economic modeling of learning. The two most popular ones are *Kalman filtering* and *recursive least squares*.[12]

As a Bayesian, the trader forecasts p_{t+1} using his posterior distribution (belief) of p_{t+1}, denoted by $f_{t+1}(p \mid x)$. $f_{t+1}(p \mid x)$ is the trader's updated subjective belief of the distribution of the price p_{t+1} after receiving the state information x at time t. The updating formula is the famous *Bayes rule*:

$$f_{t+1}(p \mid x) = \frac{f_t(p)h_t(x \mid p)}{\int f_t(\bar{p})h_t(x \mid \bar{p})d\bar{p}} \tag{15}$$

The Bayesian trader will then forecast using the posterior mean:

$$E_{b,t}(p_{t+1}) = \int p f_{t+1}(p \mid x), \tag{16}$$

where $E_{b,t}$ refers to the prediction made by the Bayesian trader at time t. Intuitively speaking, the Bayesian trader has a set of possible predictions (hypotheses) $S_t = \{p^e_{t+1}\}$, and not just a single degenerated prediction (hypothesis) p^e_{t+1}. The possibility of each of the possible predictions in the set S_t is governed by the posterior distribution (15). It is now clear that Bayesian traders need to have a greater mental capacity to first keep a set of hypotheses and then to deal with possibly very demanding computations involved in (15) and (16).[13]

There is, however, a way to reduce this very demanding work. With the assumption of the multivariate normal distribution, the entire updating of the posterior distribution can be reduced to the updating of two parameters only, namely, the mean and the variance. In this case, we have the familiar Kalman filtering. By denoting these expectations by $E_{k,t}$, then

[12] See [89] and [47] for details.

[13] It has been argued that the inability of humans to produce consistent and reliable probability and preference judgments may explain why Bayesian decision theory fails in view of this lack of necessary inputs.

$$E_{k,t}(p_{t+1}) = E_{k,t-1}(p_t) + k_t(p_t - E_{k,t-1}(p_t)), \tag{17}$$

where k_t is the *Kalman gain* at time t.

Alternatively, one can also simplify the possible messy computation by using the so-called empirical Bayes.[14] The empirical Bayesian trader basically behaves like a Bayesian, except that the posterior distribution is built upon an empirical rather than a subjective distribution. This simplification requires traders to "memorize" all of the association between the state information x and the price so that they can replace the posterior distribution $f_{t+1}(p \mid x)$ simply by using the most recent histogram. As we shall see below, this simplification connects the empirical Bayes to the K nearest neighbors, to which we now turn.

K Nearest Neighbors

KNN differs from the conventional time-series modeling techniques. The conventional time-series modeling, known as the Box-Jenkins approach, is a *global* model, which is concerned with the estimation of the function, be it linear or non-linear, in the following form:

$$p_{t+1} = f(p_t, p_{t-1}, ..., p_{t-m}) + \varepsilon_t = f(\mathbf{P_t^m}) + \varepsilon_t \tag{18}$$

by using all of the information up to t, i.e., $\mathbf{P_s^m}, \forall s \leq t$, where the estimated function $\hat{f}$ is assumed to hold for every single point in time. As a result, what will affect p_{t+1} most is its immediate past $p_t, p_{t-1}, ...$ under the law of motion estimated by all available samples.

For KNN, while what affects p_{t+1} most is also its immediate past, the law of motion is estimated *only* with *similar* samples, and *not all* samples. The estimated function $\hat{f}_t$ is hence assumed to only hold for that specific point in time. To facilitate the discussion, we introduce the following notations.

$$\mathbf{P}_1^m, \mathbf{P}_2^m, ..., \mathbf{P}_T^m, \;\; \mathbf{P}_t^m \in \mathbf{R}^m, \;\; \forall\, t = 1, 2, ..., T \tag{19}$$

$$\mathbf{P}_t^m \equiv \{p_t, p_{t-1}, ..., p_{t-m}\}, \;\; p_{t-l} \in \mathbf{R}, \forall\, l = 0, 1, ..., m-1. \tag{20}$$

$\mathbf{P}_t^m$ is a windowed series with an immediate past of m observations, also called the m-history. Equation (19), therefore, represents a sequence of T m-histories which are derived from the original time series, $\{p_t\}_{t=-m+1}^{T}$, by moving the m-long window consecutively, each with one step.

KNN forms a cluster based on each $\mathbf{P}_t^m$, $\mathscr{N}(\mathbf{P}_t^m)$, as follows.

$$\mathscr{N}(\mathbf{P}_t^m) = \{s \mid Rank(d(\mathbf{P}_t^m, \mathbf{P}_s^m)) \leq k, \forall s < t\}, \tag{21}$$

[14] See [20] for a fine overview of empirical Bayes, and also [19] for an in-depth treatment. The BUGS software provides an implementation of empirical Bayes methods using *Markov Chain Monte Carlo* [51]. The software is available from http://www.mrc-bsu.cam.ac.uk/bugs.

In other words, $\mathbf{P}_t^m$ itself serves as the centroid of a cluster, called the *neighborhood* of $\mathbf{P}_t^m$, $\mathscr{N}(\mathbf{P}_t^m)$. It then invites its *k nearest neighbors* to be the members of $\mathscr{N}(\mathbf{P}_t^m)$ by ranking the distance $d(\mathbf{P}_t^m, \mathbf{P}_s^m)$ over the entire community

$$\{\mathbf{P}_s^m \mid s < t\} \tag{22}$$

from the closest to the farthest.

Then, by assuming a functional relation, f, between p_{s+1} and $\mathbf{P}_s^m$ and using only the observations associated with $\mathscr{N}(\mathbf{P}_t^m)$ to estimate this function f_t[15], one can construct the tailor-made forecast for each p_t,

$$E_{knn}(p_{t+1}) = \hat{f}_t(\mathbf{P}_t^m). \tag{23}$$

In practice, the function f used in (23) can be very simple, either taking the *unconditional mean* or the *conditional mean*. In the case of the latter, the mean is usually assumed to be linear. In the case of the unconditional mean, one can simply use the simple average in the forecast,

$$E_{knn}(p_{t+1}) = \frac{\sum_{s \in \mathscr{N}(\mathbf{P}_t^m)} p_{s+1}}{k}, \tag{24}$$

but one can also take the weighted average based on the distance of each member. The same idea can be applied to deal with the linear conditional mean (linear regression model): we can either take the ordinal least squares or the weighted least squares. KNN can also be viewed as another kind of empirical Bayes since Equation (24) can be related to the posterior mean (16).

Embodiment: Game-Theoretic CCI

The efficient market hypothesis implies that there are no profitable strategies, and hence learning, regardless of its formalism, does not matter. As a result, the three types of traders, momentum traders, empirical Bayesian and k-nearest-neighbor traders should behave equally well, at least in the long run. However, when the market is not efficient, and learning may matter, it is expectedthat smarter agents can take advantage of dumber agents. In their experiments, [21] found that momentum traders, who never learn, performed worst during the transition period when market is not efficient. Furthermore, the empirical Bayesian traders was also outperformed by the KNN traders. While the two types of traders started learning at the same time and competed with each other to discover the true price, evidently the KNN traders were able to exploit predictability more quickly than the empirical Bayesian traders.

[21] points to a new style of application of CCI to economics, namely, using an agent-based environment to allow for a more vivid competition of different CI tools, each of which is to represent an opportunity-seeking trader with different degrees

[15] Even though the functional form is the same, the coefficients can vary depending on $\mathbf{P}_t^m$ and its resultant $\mathscr{N}(\mathbf{P}_t^m)$. So, we add a subscript t as f_t to make this time-variant property clear.

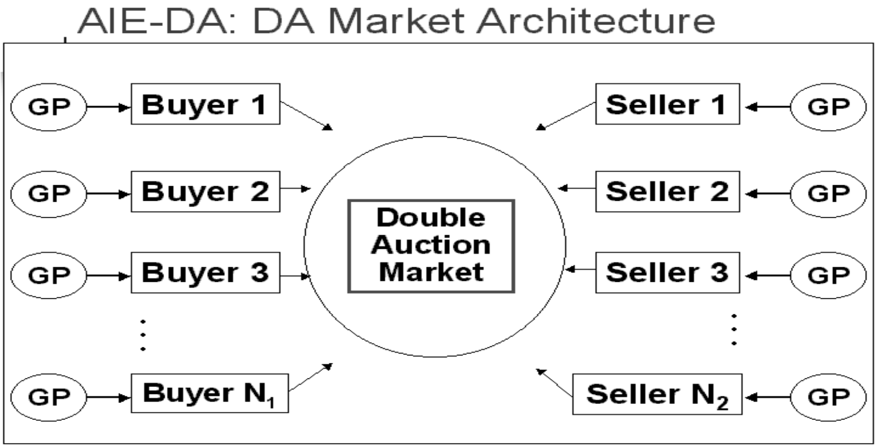


Fig. 1 Double Auction Market

of smartness. This style of application is not the same as a general forecasting tournament, in which a number of CI tools also compete with each other in forecasting a given time series. The key difference between the two styles of application lies in the *complex interacting effects* among these competing CI tools, which obviously exist in the former style, but not the latter. Put alternatively, [21] demonstrates an embodied game-theoretic environment for a set of CI tools, which may be coined as game-theoretic CCI.

2.3.3 Heterogeneity Characterized by Different Parameters

In addition to using different CI tools to characterize heterogeneity in intelligence, it is also possible to use different parameters of *the same CI tool* to distinguish different degrees of smartness. In this case, instead of the interaction of several different CI tools, what we observe is the interaction of the same CI tools, but which differ owing to different parameters.

In a study of the agent-based double auction market, [36] examine how the co-evolution of agents' strategies will change with the agents' level of smartness, and the associated micro-and-macro correspondence. Their agent-based double auction market is simulated with genetic programming (GP) (Figure 1). While GP enables all agents to learn, different values of control parameters of GP may imply different levels of smartness of agents. In this study, the parameter *population size* is used to characterize different degrees of smartness.

Using population size to characterize or approximate the degree of smartness can be justified as follows. Population size is positively associated with the search intensity. A larger search intensity can imply a higher performance.[16] In this vein, they vary their GP traders so as to have different population sizes, ranging from

[16] It is not always so. See [24].

10, 20,..., to 50. A smaller population size assumes a lower degree of smartness, whereas a larger population size implies a higher degree of smartness.

It is found that, other things being equal, increasing the intelligence of *individual traders* can contribute positively to the realized social welfare, a measure of market efficiency. Nevertheless, it is also found that, other things being equal, increasing the *number of intelligent traders* can exert an negative influence on the realized social welfare. These findings, therefore, suggest an interesting implication: if the increase in the number of intelligent traders is inevitable, then the increase in social welfare can be made possible only if all intelligent traders become smarter.

The significance of the degree of smartness is pursued in [25]. In the context of an agent-based artificial stock market, [25] address whether agents with different degrees of smartness may result in different wealth. This brings us closer to the original concerns of psychometricians mentioned in Section 2.3.1. In this study, artificial traders are all modeled by genetic algorithms (GA). They use a GA to do the forecasting, and then use it again to engage in portfolio optimization. By varying the control parameter of the GA, [25] are able to design traders with different levels of intelligence. In this case, the chosen control parameter is the *size of the validation window*, and this choice can be justified as follows.

In machine learning literature, it is very common to divide our data into three parts, namely, the training set, validation set, and testing set. The purpose of the validation sample is to prevent the trained model from being subjected to over-learning or over-fitting. In the environment of [25], it can be shown that the size of the validation window can affect the forecasting accuracy of the model constructed, which in turn will affect the quality of the portfolio decision. Through agent-based simulation, they, therefore, show that the agents' degree of smartness can positively affect their wealth share. Not surprisingly, the smarter they are, the wealthier they become.

2.3.4 Heterogeneity across Societies

Sections 2.3.2 and 2.3.3 are both concerned with an economy composed of agents with different degrees of smartness. These kinds of application can then examine how these different degrees of smartness can contribute to the resultant heterogeneity in economic performance. Therefore, they provide us with replications or predictions of the correlation between IQ and performance in a *social* context. One can further ask, to what extent, the *institutional design* can eliminate or minimize the impact of the heterogeneity in intelligence on the heterogeneity in economic performance, for example, income inequality [79].

However, one may also be concerned with the comparisons between different economies or different groups of agents. For example, [72, 71] provide rich resources on the comparative studies of IQ among different countries and races. On the other hand, differences in individuals' behavior among different societies can also be attributed to the *culture* factor. The recent path-breaking studies in this area can be found in [55, 56]. Using experimental results from the *ultimatum bargaining games*, [55] is able to show that “economic decisions and economic reasoning may be heavily influenced by cultural differences–that is, by socially transmitted rules about how to behave in certain circumstances (economic or otherwise) that may vary from group to group as a consequence of different cultural evolutionary trajectories." (Ibid, p. 973)

With these backgrounds, it is useful to distinguish different groups of agents by using different CI tools. For example, one can use reinforcement learning to model a group of agents, and use genetic algorithms to model another group of agents. One may also try the same CI tool but with different control parameters to characterize two different societies, as what we do in Section 2.3.3. Then by putting these two groups of agents separately in the same environment, such as the ultimatum bargaining game or stock market, one can compare the performance of the two different groups. In this way, the possible impacts of genetics and cultures upon the collective performance can be tackled. This research direction can be exemplified by the following few examples.

Minority Games

[90] addressed the traffic-flow problem in the context of games. This issue concerns the most efficient distribution of the road space among drivers, characterized by the travel time among different paths among drivers connecting the same origin with the destination being identical. The intriguing part of this issue is: can we achieve this goal in a bottom-up manner without the top-down supervision? [90] explored the possibility by assuming that each driver learns how to choose the paths by means of reinforcement learning. Several different versions of reinforcement learning have been attempted. They differ in one key parameter, *learning speed* or *the degree of forgetting*. It has been found that the allocative efficiency of roads is not independent of this parameter. In other words, unless the learning speed is tuned correctly, there is no guarantee that drivers will necessarily coordinate their use of roads in the most efficient way, and congestion can happen all the time.

The congestion problem, also known as *minority games*, originates from the famous *El Farol problem*, which was first studied by [9]. The problem concerns the attendance at the bar, El Farol, in Sante Fe. Agents' willingness to visit the bar on a specific night depends on their expectations of the attendance at that night. An agent will go to the bar if her expected attendance is less than the capacity of the bar; otherwise, she will not go. [9] showed that the time series of attendance levels seems to always fluctuate around an average level of the capacity of the bar. However, agents in [9] reason with *a fixed set of models*, deterministically iterated over time. Discovering new models is out of the question in this set-up.

[49] replace this fixed set of rules with a class of *auto-regressive* (AR) models. Furthermore, the number of lag terms and the respective coefficients are revised and renewed via *evolutionary programming* (EP). The introduction of EP to the system of AR agents has a marked impact on the observed behavior: the overall result is one of large oscillations rather than mild fluctuations around the capacity.

[90] and [49] together show that there is no guarantee that agents with *arbitrary learning algorithms*, characterizing different cultures, habits, routines, or IQ, can

coordinate well to avoid congestion and maximize social efficiency, and the *coordination limit* is affected by the IQ or cultures of society, which are characterized by various computational intelligence tools.

2.4 Molecule Level: Hybridization

The idea of hybrid systems is also employed to build individual agents. In this case, each individual is represented by more than one CI tool, and is an incarnation of a specific style of CCI.

Evolutionary Artificial Neural Nets

[67] provides the first application of this kind, and, in this case, the specific style is the *evolutionary artificial neural net*. In the context of the SFI artificial stock market, the financial agents are required to solve the portfolio optimization problem, which involves the distribution of the savings into risky and riskless assets, something which is similar to Equation (11). Equation (11) is a typical two-stage decision, i.e., the forecast decision is made before the investment decision, but [67] considered a reduced one-stage decision. The mapping is, therefore, directly constructed from the information available at time $t-1$ to the optimal portfolio at time t, $y_{h,t}$. More precisely, the financial agents are first represented by an artificial neural network, or a feedforward neural network with one hidden layer, to be exact.

$$y_{h,t} = h_2(w_0 + \sum_{j=1}^{l} w_j h_1(w_{0j} + \sum_{i=1}^{p} w_{ij} x_{i,t-1})) \tag{25}$$

The information set, $\{x_i\}_{i=1}^{p}$ includes past dividends, returns, the price/dividend ratio, and trend-following technical trading indicators. This population of investment decision rules (over all agents) is then evolved with *genetic algorithms* to symbolize the evolutionary learning of financial agents.

Genetic Fuzzy Classifier System

Another related development has occurred in the use of *natural language*. People frequently and routinely use natural language or linguistic values, such as high, low, and so on, to describe their perceptions, demands, expectations, and decisions. Some psychologists have argued that our ability to process information efficiently is the outcome of applying *fuzzy logic* as part of our thought processes. The evidence on human reasoning and human thought processes supports the hypothesis that at least some categories of human thought are definitely fuzzy. Yet, early agent-based economic models have assumed that agents' adaptive behavior is *crisp*. [98] made progress in this direction by using the *genetic-fuzzy classifier system* (GFCS) to model traders' adaptive behavior in the SFI-like artificial stock market.

[98] considers a fuzzy extension of the forecasting function (13). In Equation (13), each forecasting rule has two coefficients, the constant term (α) and the

slope (β). Without any augmentation, these forecasting rules are simply linear, and cannot be expected to work well. The original SFI-ASM made them non-linear by making these two coefficients *state dependent* via the classifier system. However, the two coefficients are crisp. [98] applies the Mamdani style of fuzzy rules to make them fuzzy. As an illustration, the Mamdani style of a fuzzy if-then rule is:

If x is "A", then y is "B",

whereas the input set "A" and the output set "B" are both fuzzy. In [98], this application becomes something like:

If $\frac{p_t}{MA(5)}$ is " low", then α is "moderately high", and β is "moderately high".

Obviously, the terms "low", "high", "moderately low", and "moderately high" are all linguistic variables, and they are represented by the respective membership functions. The state variable is $p_t/MA(5)$, where $MA(5)$ is the moving average of the price over the last five periods. So, this rule compares the current price with the 5-day moving average, and if the ratio is low enough, then both α and β will be moderately high. Of course, the above fuzzy forecasting rule can easily be extended to include more variables. For example,

"If $\frac{p_t \times r_t}{d_t}$ is high and $p_t/MA(5)$ is moderately low and $p_t/MA(10)$ is moderately high and $p_t/MA(100)$ is low and $p_t/MA(500)$ is high, then α is "moderately low", and β is "high".

$p_t r_t/d_t$ reflects the current price in relation to the current dividend and it indicates whether the stock is above or below the fundamental value at the current price. The inclusion of this information makes agents behave like fundamentalists. The remaining four state variables indicate whether the price history exhibits a trend or similar characteristic. The inclusion of this information makes agents behave more like chartists. Therefore, by combining these state variables, the financial agents may choose to behave more like fundamentalists or more like chartists.[17]

3 Human and Software Agents

We have now reviewed how the idea of CCI enhances the heterogeneous-agent research paradigm at the macro, micro and molecule levels. In addition to that, the idea of CCI also plays an important role in the recent efforts made by economists to overarch agent-based computational economics (ACE) and experimental economics. It has been argued in many instances that agent-based simulation should be integrated with experiments using human subjects, for example, [45], [60] and [74]. The relationship between agent-based computational economics and experimental

[17] This design is not the 2-type design as we see in Section 2.2.2.

economics is, in essence, a relationship between *human agents* and *software agents*. The literature has already demonstrated three possible ways of closely relating ACE to experimental economics, namely, *mirroring*, *competition* and *collaboration*. They appear in the literature in chronological order.

3.1 Mirroring

The early ACE studies are clearly motivated by using software agents to mimic the behavior of human agents observed in the laboratory. The famous *Turing test* serves as the best illustration. [8] point out that the development of social science theories can be likened to the task of building a computer to mimic human behavior, or equivalently, to building a computer that will pass the Turing test in the range of behavior covered by the theory. Thus, a social science theory can be deemed to be successful when it is no longer possible for a computer judge to tell the difference between behavior generated by humans and that generated by the theory (i.e., by a machine).

3.1.1 Mirroring with Genetic Algorithms

In this regard, the two CI tools, namely, genetic algorithms and genetic programming are frequently used to build software agents such that their collective behavior can mirror the laboratories with human subjects. [6] pioneered this research direction. [6] applied two versions of GAs to study market dynamics in a *cobweb model*. The basic GA involves three genetic operators: reproduction, crossover, and mutation. Arifovic found that in each simulation of the basic GA, individual quantities and prices exhibited fluctuations for its entire duration and did not result in convergence to the rational expectations equilibrium values, which is quite inconsistent with experimental results involving human subjects.

Arifovic's second GA version, the augmented GA, includes the election operator in addition to reproduction, crossover, and mutation. The election operator involves two steps. First, crossover is performed. Second, the potential fitness of the newly-generated offspring is compared with the actual fitness values of its parents. Among the two offspring and two parents, the two highest fitness individuals are then chosen. The purpose of this operator is to overcome difficulties related to the way mutation influences the convergence process, because the election operator can bring the variance of the population rules to zero as the algorithm converges to the equilibrium values.

The results of the simulations show that the augmented GA converges to the rational expectations equilibrium values for all sets of cobweb model parameter values, including both stable and unstable cases, and can capture several features of the experimental behavior of human subjects better than other simple learning algorithms. To avoid the arbitrariness of choice of an adaptive scheme, [70] suggested that comparison of the behavior of adaptive schemes with behavior observed in laboratory experiments with human subjects can facilitate the choice of a particular adaptive scheme. From this suggestion, the GA could be considered an appropriate choice to model learning agents in a complex system. Other similar studies to justify the use of genetic algorithms to mirror human experiments include [7].

3.1.2 Mirroring with Genetic Programming

The application of genetic programming to the cobweb model started from [30]. [30] compared the learning performance of GP-based learning agents with that of GA-based learning agents. They found that, like GA-based learning agents, GP-based learning agents can also learn the homogeneous rational expectations equilibrium price under both the stable and unstable cobweb case. However, the phenomenon of price euphoria, which did not happen in [6], does show up quite often in the early stages of the GP experiments. This is mainly because agents in their setup were initially endowed with very limited information as compared to [6]. Nevertheless, GP-based learning can quickly coordinate agents' beliefs so that the emergence of price euphoria is only temporary. Furthermore, unlike [6], [30] did not use the election operator. Without the election operator, the rational expectations equilibrium is exposed to potentially persistent perturbations due to the agents' adoption of the new, but untested, rules. However, what shows up in [30] is that the market can still bring any price deviation back to equilibrium. Therefore, the self-stabilizing feature of the market, known as the invisible hand, is more powerfully replicated in their GP-based artificial market.

The self-stabilizing feature of the market demonstrated in [30] was further tested with two complications. In the first case, [31] introduced a population of speculators to the market and examined the effect of speculations on market stability. In the second case, the market was perturbed with a structural change characterized by a shift in the demand curve, and [32] then tested whether the market could restore the rational expectations equilibrium. The answer to the first experiment is generally negative, i.e., speculators do not enhance the stability of the market. On the contrary, they do destabilize the market. Only in special cases when trading regulations, such as the transaction cost and position limit, were tightly imposed could speculators enhance the market stability. The answer for the second experiment is, however, positive. [32] showed that GP-based adaptive agents could detect the shift in the demand curve and adapt to it. Nonetheless, the transition phase was non-linear and non-smooth; one can observe slumps, crashes, and bursts in the transition phase. In addition, the transition speed is uncertain. It could be fast, but could be slow as well.

In addition to genetic algorithms, genetic programming is also extensively applied to build systems of software agents which are able to replicate the laboratory results with human subjects. [26] studied bargaining behavior observed in the double-auction laboratory markets with human subjects. All buyers and sellers in [26] are *artificial adaptive agents*. Each artificial adaptive agent is built upon *genetic programming*. The architecture of genetic programming used is what is known as *multi-population genetic programming* (**MGP**). Briefly, they viewed or modeled

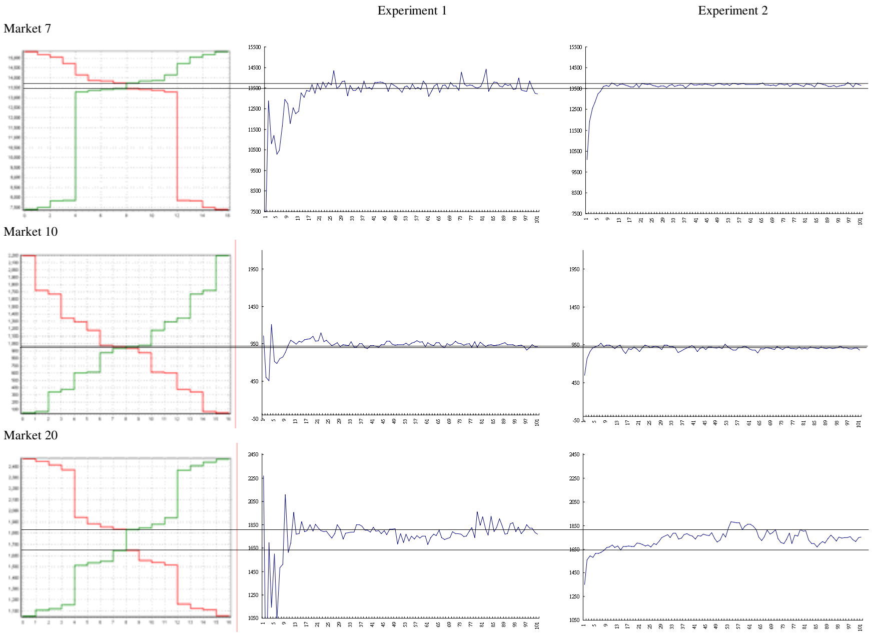


Fig. 2 Agent-Based Double Auction Market Simulation with GP Agents. The three markets presented here are selected and adapted from [26], Fig. 4

an agent as *a population of bargaining strategies*.[18] Genetic programming is then applied to *evolving* each population of bargaining strategies. In this case, a society of bargaining agents consists of many populations of programs. This architecture is shown in Figure 1.

In Figure 2, there are three plots. The leftmost plot is the market with its equilibrium price or equilibrium price interval. The middle plot and the rightmost plot are the time series of the GP simulations with different parameters. As we can see from Figure 2, the three markets presented here either have a unique equilibrium price (Market 10) or a tight equilibrium interval (Markets 7 and 20). Market prices in this case quickly move toward the equilibrium price (or price interval), and then slightly fluctuate around there. This result is basically consistent with what we learned from experimental economics with human subjects [96].

3.2 *Competition*

In addition to mirroring the collective behavior of human agents, software agents are also used directly to interact with human agents. This advancement is partially

[18] The number of bargaining strategies assigned to each bargaining agent is called the *population size*. **AIE-DA Version 2**, developed by the AI-ECON Research Center, allows each agent to have at most 1000 bargaining strategies.

motivated by the increasing popularity of electronic commerce. In web-based on-line markets, such as *ebay*, one general question concerns the role of software agents. Can software agents perform better than human agents in making deals? This question is similar to the inquiry on the degree of smartness of software agents in comparison with that of human agents, which is a generalization of the issue pursued in Section 2.3.1. Of course, it would be problematic to measure the IQ of software agents. Nevertheless, the Turing test does challenge the possibility that one can measure the IQ of software agents if they are properly designed. As a result, as an extension of Section 2.3.2, we can now match human agents with software agents in various kinds of markets or games.

The agent-based financial system **U-MART** provides one of the best illustrations. U-MART stands for *UnReal Market as an Artificial Research Test bed*. This is a research project collaboratively initiated by a number of universities in Japan [95]. It is an agent-based future market, which serves both purposes of education and research. Second, U-MART enables us to address a very basic question: can human agents dominate the software agents when they are placed together in the market? What was found in some limited experiments is that the performance of software agents was superior to that of human agents. Of course, the result is still premature. Even the question is not well-defined. This is so because the competition was generally not made on a fair basis. For example, during the transaction process, human agents were poorly equipped with computational facilities so that they were not able to figure out some important figures in the nick of time. The competition will be considered more fair if it allows human agents to write their own trading programs and modify the program on-line, as will be discussed in Section 3.3.

Other interesting research questions can also be addressed by watching the interaction between human and software agents. Market efficiency is a case in point. [53] studied whether market efficiency can be enhanced when software agents are introduced to the markets which were originally composed solely of human agents.[19] They designed a continuous double auction market in the style of the Iowa electronic market, and introduced software agents with a passive arbitrage seeking strategy to the market experiment with human agents. They then went further to distinguish the case where human agents are informed of the presence of software agents from the case where human agents are not informed of this presence. Whether or not the human agents are well informed of the presence of the software agents can have significant impacts upon market efficiency (in the form of price deviations from the fundamental price). They found that if human agents are well informed, then the presence of software agents triggers more efficient market prices when compared to the baseline treatment without software agents. Otherwise, the introduction of software agents results in lower market efficiency.[20] In a sense, this question can be viewed in terms of the *socio-psychological impact* on human behavior in the presence of interacting machine intelligence.

[19] See also [91] and [101].

[20] These issues have been further pursued in the recent development of the **U-Mart** platform ([99], [92], [65]).

3.3 Collaboration

At the third stage, neither do we mirror, nor do we match, the two kinds of agents. Human agents now work with software agents as a *team*, and they are no longer treated as two entities. The modern definition of artificial intelligence has already given up the dream of passing the Turing test. Instead, a more realistic and also interesting definition is based on the team work cooperatively performed by software agents and human agents [68]. If the studies of the first two stages can be considered to be the works under the influence of classical AI, then the third development is a natural consequence of the modern AI.

3.3.1 Is Collaboration Behaviorally Feasible?

This stage of research involves one non-trivial question: *is collaboration behaviorally feasible?* Would human agents choose to work with software agents if they were given this option? Can we see the collaborative computational intelligence beyond software agents in this way? [28] carried out an experiment to make software agents available in a laboratory with human subjects, and to watch whether human agents will choose software agents to be their representatives, while forming their decisions and actions. The laboratory used to facilitate this study is a *prediction market*, which is a kind of web-based and agent-based simulation platform [102]. They designed the software agents as the trading algorithms which can execute bid orders or ask orders when the market timing condition is met. Human agents have the option of choosing these software agents, and even rewrite these algorithms, to be their trading representatives. If they do so, their identities are completely characterized by the associated software agents. In this case, the distinction between human and software agents becomes problematic.

They carried out the above experiment in relation to a recent political event (a mayoral election in Taiwan), and found that out of 425 market participants (human agents), only 25 used software agents to trade. In other words, most human agents chose not to bother with software agents. This result is evidence that human agents may not collaborate with software agents. They then went further to ask the rationale behind the choice to ignore software agents. They first found that human agents who used software agents to trade generally earned higher profits than those who did not use them, and this dominance was statistically significant. This brought us face to face with a puzzling question: Why do human agents not “recruit” software agents when they are available in a promising way? To tackle this question, they distributed questionnaires to market participants to gain more information regarding their choice behavior.

3.3.2 Collaborate with Customized Software Agents

One of the reasons why human agents did not use software agents as their representatives was that they did not feel comfortable with them, or else they did not quite trust them. In other words, these software agents were not customized. In a subsequent experiment, they considered a different setup which may blur the relations between human and software agents. That is, they asked each human agent to write his or her own trading program (software agents), and used them as their incarnations in the later agent-based market simulation. This idea is very similar to the game-like tournaments pioneered by Robert Axelrod in the mid-1980s [12, 63], and the market-like tournament initiated by the Santa Fe Institute in the early 1990s [86]. However, moving one step further, they considered a comparison between the simulation based on these humanly-supplied software agents and the one based on purely computer-generated software agents. For the latter case, genetic programming was applied to generate the autonomous agents, and the platform **AIE-DA** was used to implement the simulation (see also Section 3.1.2).

Out of the 20 simulations which they carried out, they found that the market composed of purely computer-generated software agents that are autonomous and adaptive performs consistently better, in terms of market efficiency, than the market composed of humanly-supplied software agents, even though humanly-supplied software agents are more sophisticated or thoughtful.

The two experiments above together have two implications. The first experiment, from a sociological viewpoint, provides evidence that human agents may have difficulty embracing (containing) software agents when making decisions. The second experiment further shows that if we allow human agents to choose or even design their own software agents, their collective behavior may not be the same as that observed from the computer-generated software agents. This second finding is similar to that of [8].[21]

4 Hybrid Systems

Section 2 reviews the applications of CCI to *agent-based computational economics* (the heterogeneous-agent research paradigm), and Section 3 reviews the applications of CCI to *experimental economics*. In both of these two cases, CCI is mainly used to build software agents in economic and financial models. In other words, CCI contributes to *economic and financial agent engineering*. Another major area to which CCI is also vastly applied, with an even longer history, is *financial engineering*. This application is mainly motivated by the rapid development of various hybridizations of CI tools. There are a number of hybrid systems frequently observed in financial engineering. We shall only sample some in this section.

4.1 Nature of Hybridization

The idea of using various CI tools as a *module* of a hybrid system or an *agent* of a multi-agent system is not new, but it has just gained its momentum over the last

[21] [8] find significant differences in data generated by some chosen learning models and humans, with the greatest ones in coordination games.

decade. [22] The room for this collaboration is available mainly because of the heterogeneity of CI tools (see also Section 2). Different CI tools are designed with different mechanisms inspired from various natural and artificial processes; therefore, they may each handle one or a few different aspects of an intelligent task [23]. To name a few, self-organizing maps are operated for pattern discovery and concept formation, auto-association neural networks are good for the removal of redundancies and data compression, feedforward and recurrent neural networks are regarded as universal function approximators, and the approximation process can usually be facilitated by genetic algorithms. With this diversity in specialization, it would be surprising to see very little collaboration but much competition among them, as has been developed in the literature over the past. The recent research trend seems to recognize this biased development and move back to the collaboration theme accordingly.

Financial hybrid systems are mainly the financial applications of the hybrid systems or multi-agent systems. Among the many designs of financial hybrid systems, it is important to distinguish *models* from *processes*, particularly, *evolutionary processes*. Many financial hybrid systems are designed based on the idea of putting a model or a population of models under an evolutionary process, which includes evolutionary artificial neural networks, evolutionary fuzzy inference systems, evolutionary support vector machines, etc. We shall start with a review of this main idea (Section 4.2). The second major element we experienced in hybrid financial systems is the desire to make semantic sense of the hybrid systems, which includes the use of natural language and qualitative (non-numeric) reasoning. We shall then provide a brief review of this (Section 4.3). We conclude this section by mentioning the collaboration work done with the data or database, such as feature selection, dimension reduction, etc. (Section 4.4).

4.2 Evolutionary-Based Hybridization

The idea of the evolutionary-based hybridization is clear: there are two major elements in this hybrid system. One is the *universal function approximator*, and the other is the *built-in approximation process*. If we consider the former as a parametric model, then the latter can be regarded as an estimator of it. In economics, there are three frequently used evolutionary-based hybrid systems, namely, evolutionary artificial neural networks (Section 4.2.1), evolutionary support vector machines (Section 4.2.2), and evolutionary fuzzy or neurofuzzy inference systems.

4.2.1 Evolutionary Artificial Neural Networks

Among all hybridizations of CI in finance, the most popular one is probably the combination of genetic algorithms and artificial neural nets, which is one of the kinds of *evolutionary artificial neural nets* (EANNs), referred as to GANNs (genetic

[22] A more comprehensive treatment of the hybrid system can be found in [18] and [41].

algorithms + neural nets). EANNs are computationally very demanding; therefore, despite their conceptual appeal, there were few financial applications using EANNs in the 1990s. However, recently, a number of commercial algorithms, such as *NeuroSolutions*, have been made available, and hence the adoption of EANNs has spread quickly and widely.

In GANNs, genetic algorithms are applied to evolve an artificial neural net so as to genetically determine not only its weights, but also its topology, including the number of layers, the number of hidden nodes and input selection. For example, [64] uses a genetic algorithm to simultaneously optimize the connection weights between layers and select instance. It is found that genetically selected instances shorten the learning time and enhance prediction performance. It many applications, the weight determination is performed through the backpropagation algorithm, but in this case genetic algorithms can be applied to determine the learning rate and momentum of the backpropagation algorithm [94].

In addition to GAs, another subclass of EANNs is GPNNs (genetic programming + neural nets). [3] employ genetic programming to evolve artificial neural networks, and the genetically evolved neural networks are applied to forecast exchange rate returns for the Japanese Yen and the British Pound against the US dollar. The empirical results show the existence of a short-term weak predictable structure for both currencies.

Not being confined to feedforward neural nets, GAs have also been used to evolve other kinds of neural nets. [94] use a GA to genetically evolve recurrent neural networks. In this study, a hybrid model that combines a seasonal linear time series model and a recurrent neural network is used to forecast agricultural commodity prices. A genetic algorithm is employed to determine the optimal architecture of the ANNs. It turns out that the out-of-sample prediction can be improved slightly with the hybrid model. [105] uses a GA to evolve *fuzzy neural networks* for financial forecasting. In this study, the genetic algorithm and the gradient decent (GD) algorithm are used alternatively in an iterative manner to optimize all parameters and weights in a five-layer fuzzy neural network. It is found that the hybrid learning algorithm combining GA and GD is more powerful than the previous separate sequential trading algorithm.

The most active financial domain using EANNs is financial time series forecasting. [83] apply genetically evolved neural network models to predict the Straits Times Index (STI) of the Stock Exchange of Singapore. This study shows that satisfactory results can be achieved when applying genetically evolved neural networks to predict the STI. EANNs' applications to other financial domains include *financial distress prediction* [78] and *trading* [93, 14]. [93] enhance EANNs with fractal analyses. Based on Hurst exponent calculations, the appropriate input windows for the EANN are identified. It then investigates the efficacy of the model using closing price time series for a suite of stocks listed on the SPI index on the Australian Stock Exchange. The results show that Hurst exponent configured models out-perform basic EANN models in terms of average trading profit found using a simple trading strategy.

4.2.2 Evolutionary Support Vector Machines

In the 1990s, based on results from *statistical learning theory*, an alternative to the artificial neural network was developed, i.e. the support vector machine (SVM), also called the *kernel machine*. It has been found that when compared with the standard feedforward neural nets trained with the backpropagation algorithm, support vector machine can have superiorperformance [37]. This superiority may be attributable to different optimization principles running behind the two: for the SVM, it is the *structural risk minimization principle*, whereas for backpropagation it is the *empirical risk minimization principle*. The objective of the former is to minimize the upper bound of the generalization error, whereas the objective of the latter is to minimize the error based on training data. Hence, the former may lead to better generalization than the latter. Partially due to this advantage, the financial applications of SVM have kept on expanding.[23]

However, like the ANN, the SVM can also be treated as a semi-parametric model. To use it, there are a number of parameters or specifications that need to be determined. Basically, there are three major parameters in the SVM. First, there is the penalty parameter associated with the empirical risk appearing in the structural risk function. In the literature, it is normally denoted by C. Second, when the SVM is applied to the regression problem, in addition to C, there is a parameter ε appearing in the ε-error intensive function. Third, it is the parameter of the chosen kernel function. Support vector machines non-linearly map a lower dimensional input space into a high dimensional, possibly, an infinite dimensional, feature space. However, a central concept of the SVM is that one does not need to consider the feature space in explicit form; instead, based on the Hilbert-Schmidt theory, one can use the kernel function. There are two kernel functions frequently used, namely, the *Gaussian kernel* and the *polynomial kernel*. The former has a parameter associated with the second moment of a Gaussian called *width* (normally denoted by σ), and the latter has a parameter associated with the polynomial degree (normally denoted by p).

At the beginning, these parameters were arbitrarily given by trial-and-error. Later on, genetic algorithms were extensively employed to optimize the SVM, and the applications of ESVM have been seen in various fields, such as reliability forecasting [81], traffic flow forecasting [97], bankruptcy forecasting [1, 103, 106], and stock market prediction [107, 40].

[103] uses a GA to genetically determine the parameters C and σ of the SVM. The proposed GA-SVM was then tested on the prediction of financial crisis in Taiwan. The experimental results show that the GA-SVM model performs better than the manually-driven SVM. [1] use a GA to simultaneously optimize the feature selection and the instance selection as well as the parameters of a kernel function. It is also found in this study that the prediction accuracy of the conventional SVM may be improved significantly by using the ESVM. In [107] a GA is used for variable selection in order to reduce the modeling complexity of the SVM and improve the

[23] The interested reader can find some useful references directly from the website of the SVM: http://www.svms.org/

speed of the SVM, and then the SVM is used to identify the stock market movement direction based on historical data.

4.3 Semantics-Based Hybrid Systems

Fuzzy-based modeling is appealing for social scientists because the use of linguistic variables enables them to make semantic sense of their models, which generally leads to easy interpretation of the rules or knowledge extracted from the models. By contrast, many so-called “black-box” CI tools are often criticized for the lack of this property. Therefore, fuzziness becomes an imperative element for building intelligent systems, and, like an evolutionary mechanism, its necessity generates another series of hybrid systems, that we generally call *semantics-based hybrid systems*. From our point of view easy to understand is the distinguishing feature of semantics-based hybrid systems, and among all economic and financial applications of them, *neuro-fuzzy systems* are probably the most popular ones.

A neuro-fuzzy system is a fuzzy system that is represented by a kind of neural network, for example, the feedforward neural network, and, therefore, it can be trained (estimated) by using the associated learning algorithms of the network, e.g., backpropagation. Therefore, the neuro-fuzzy system is a *learning fuzzy system* in which the neural network learning algorithms are used to determine parameters of the fuzzy system. Several different neuro-fuzzy systems exist in the literature [80], but only the *ANFIS* (Adaptive Network-based Fuzzy Inference System) is widely used in economic and financial applications.

4.3.1 ANFIS

The ANFIS was proposed by [59]. It represents a *Sugeno-style* fuzzy system in a special five-layer feedforward neural network. The Sugeno style of a fuzzy if-then rule is:

If x is “A” and “y” is “B”, then $z = f(x,y)$.

“A” and “B” above are fuzzy sets. However, the function $f(x,y)$ in the ANFIS is linear:

$$z = f(x,y) = \alpha + \beta_x x + \beta_y y. \tag{26}$$

The structure can, therefore, be comparable to the autonomous-agent design in the SFI artificial stock market (see Section 2.2.3, Equation 13) and is even closer to the modified version of the SFI-ASM proposed by [98] (see Section 2.4). However, unlike [98], the rule base used in the ANFIS must be known in advance. The ANFIS integrates the backpropagation algorithm with the recursive least squares algorithm to adjust parameters.

The ANFIS has been applied to water consumption forecasting [11], stock prices forecasting [13, 39, 108], credit scoring [73], market timing decisions [38], credit risk evaluation [104], and option pricing [62].

[62] applies the ANFIS to option market pricing based on the transaction data of the Indian Stock Option. The pricing capability of the ANFIS is compared with the performance of the ANN model and Black-Scholes (BS) model. The empirical results show that the out-of-sample pricing performance of the ANFIS is superior to that of the BS, and is also better than the ANN. In addition, compared to the ANN, the ANFIS is explicit about its decision rules.

Instead of backpropagation, [39] uses *extended Kalman filtering* to estimate the ANFIS, and demonstrates its performance by comparing it with the ANFIS with regard to stock index forecasting. It is found that the proposed extended Kalman filtering can perform better than backpropagation.

4.3.2 Other Neuro-fuzzy Systems

In addition to the ANFIS, financial applications using other neuro-fuzzy systems also exist. [46] present a cooperative neuro-fuzzy inference system to forecast the expected financial performance of farm businesses. The fuzzy inference system considered is the *Mamdani Style* rather than the Sugeno Style generally used in the ANFIS. In addition, [46] only estimates the rule weight, and the parameters of the membership function, which is a sigmoid function, are not part of it. Therefore, it uses Kosko's FAM (*fuzzy associative memories*) vector quantization algorithm (or competitive learning algorithm) [66] to estimate the fuzzy inference system. The proposed system is compared with the conventional ordered multinomial logit regression model. The result shows that logit regression generally classified farms more accurately, but the FAM model was more accurate at predicting poorly performing farms, and, more importantly, the development and interpretation of the NFIS was found to be very intuitive.

[100] propose a new neural fuzzy system, namely the *generic self-organizing fuzzy neural network* based on the *compositional rule of inference*, as an alternative to predict banking failure. The system referred to as GenSo is able to identify the inherent traits of financial distress based on financial features derived from publicly available financial statements. The interaction between the selected features is captured in the form of highly intuitive if-then fuzzy rules. Such rules hence provide insights into the possible characteristics of financial distress and form the knowledge base for a highly desired early warning system that aids bank regulation.

4.4 Feature Reduction: Rough GA or GP

The hybrid system which we shall review in this section comprises two CI tools, namely, genetic programming and rough sets. The hybridization of GP and rough sets provides an excellent illustration of how the usual competitive relationship between two CI tools can be more productively transformed into into a collaborative relationship ([77, 88]).

Rough sets define a mathematical model of vague concepts that is used to represent the relationship of dissimilarity between objects. Two objects are considered *equivalent* with respect to *a certain subset of attributes* if they share the same value for each attribute of the subset. By collecting all equivalent objects, one can decompose the entire universal (set of objects) into equivalent classes. The decomposition, of course, is not unique and is dependent on the subset of attributes which we use to define the equivalent relation.

Rough sets arise when one tries to use the equivalent classes with respect to some attributes to give a description of a concept based on the associated decision attributes. For example, if the decision attribute concerns financial distress and is binary (bankruptcy or not), then what one wants to characterize is the concept of *bankruptcy* by using some attribute of the firms, e.g., their financial ratios, size, etc. The characterizations are only approximate when complete specification of the concept is infeasible. In this case, the concept itself is rough, and the objects associated with the concept are referred to as the rough set.

Two partial specifications are considered to be the most important, namely, the *lower approximations* and the *upper approximations*. The lower approximations consist of objects (equivalent classes) which belong to a concept with certainty, i.e., the entire equivalent classes are a subset of the rough set. The upper approximations consist of those equivalent classes which possibly belong, i.e., they have non-empty intersection with the rough set. A subset of the attributes is called *reduct* if all attributes belonging to it are *indispensable*. An attribute is *dispensable* if its absence does not change the set approximation. In other words, a reduct is a set of attributes that is sufficient to describe a concept.

A financial hybrid system using rough sets and GP is first proposed by [77]. In this hybridization, the rough set is firstly adopted to select the discriminative features by identifying *reducts*. Only these reducts are then taken as the input features for the GP learning process. [77] uses genetic programming to construct a bankruptcy prediction model with variables from a rough sets model. The genetic programming model reveals relationships between variables that are not apparent in using the rough sets model alone.[24]

5 Concluding Remarks

According to the current trend in the literature, this paper addresses what collaborative computational intelligence can mean for economists. While the recent series of publications on the economic and financial applications of computational intelligence has already demonstrated the relevance of various CI tools to economists [29, 35], they are mostly taken as techniques for economists. In this chapter, we go

[24] There are many other ways to hybridize GP or GA with rough sets, but so far their financial applications have rarely been seen.

one step further to show that they can be more productive so as to be part of the future of economics. Specifically, we demonstrate this potential by singling out two new research paradigms in economics, namely, agent-based economics and experimental economics.

The essence of agent-based economics is a society of heterogeneous agents, a subject which is highly interdisciplinary motivated. Collaborative computation intelligence enables or inspires economists to see how some initial explorations of the richness of this society can be made. In this regard, computational intelligence may contribute by providing not just models of learning or adaptation, but models of learning or adaptation processes which may be influenced by behavioral genetic and cultural factors.

After one decade of rapid development, a challenging issue facing experimental economics is how to strengthen the reliability of the laboratory results with human subjects by properly introducing software agents to labs. In fact, the state-of-art economic laboratory is no longer a lab with only human subjects, but a lab comprising both human agents and software agents [27]. Collaborative computational intelligence can contribute significantly to the building of the modern lab.

The last part of the paper reviews some recent economic and financial applications of hybrid systems. However, there is no attempt to give an exhaustive list, which itself may deserve a separate treatment. We, therefore, single out the two most significant elements in frequently used economic and financial hybrid systems, namely, evolution and semantics. The former mainly contributes to the hybrid system as a process to facilitate the universal approximation, whereas the latter contributes to the hybrid system by enhancing its semantics.

To sum up, this chapter has shown how collaborative computational intelligence has enriched the design of economic and financial agents, while, in the meantime, providing quantitative economists with a longer list of ideas to cope with the inherent complexity and uncertainty in the data.

Acknowledgements. The author is grateful to one anonymous referee for his helpful suggestion. The author is also grateful to the editor, Dr. Christine L. Mumford, for her devoted efforts made in editing the chapter. NSC research grant No. 95-2415-H-004-002-MY3 is gratefully acknowledged.

References

1. Ahn, H., Lee, K., Kim, K.-J.: Global optimization of support vector machines using genetic algorithms for bankruptcy prediction. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4234, pp. 420–429. Springer, Heidelberg (2006)
2. Allen, H., Taylor, M.: Charts, noise and fundamentals in the London foreign exchange market. Economic Journal 100, 49–59 (1990)
3. Alvarez-Diaz, M., Alvarez, A.: Forecasting exchange rates using an evolutionary neural network. Applied Financial Economics Letters 3(1), 5–9 (2007)
4. Amilon, H.: Estimation of an adaptive stock market model with heterogeneous agents. Journal of Empirical Finance (forthcoming) (2008)

5. Aoki, M., Yoshikawa, H.: Reconstructing macroeconomics: A perspective from statistical physics and combinatorial stochastic processes. Cambridge University Press, Cambridge (2006)
6. Arifovic, J.: Genetic algorithm learning and the cobweb model. Journal of Economic Dynamics and Control 18(1), 3–28 (1994)
7. Arifovic, J.: Genetic algorithms and inflationary economies. Journal of Monetary Economics 36(1), 219–243 (1995)
8. Arifovic, J., McKelvey, R., Pevnitskaya, S.: An initial implementation of the Turing tournament to learning in repeated two-person games. Games and Economic Behavior 57(1), 93–122 (2006)
9. Arthur, W.: Inductive reasoning and bounded rationality. In: American Economic Association Papers Proceedings, vol. 84, pp. 406–411 (1994)
10. Arthur, W., Holland, J., LeBaron, B., Palmer, R., Tayler, P.: Asset pricing under endogenous expectations in an artificial stock market. In: Arthur, B., Durlauf, S., Lane, D. (eds.) The economy as an evolving complex system II, pp. 15–44. Addison-Wesley, Reading (1997)
11. Atsalakis, G., Ucenic, C.: Time series prediction of water consumption using the neurofuzzy (ANFIS) approach. In: IWA International Conference on Water Economics, Statistics, and Finance, Rethymno, Greece, July 8–10, vol. I, pp. 93–100 (2005)
12. Axelrod, R.: The evolution of cooperation. Basic Books (1984)
13. Azeem, M., Hanmandlu, M., Ahmad, N.: Generalization of adaptive neuro-fuzzy inference systems. IEEE Transactions on Neural Networks 11(6), 1332–1346 (2000)
14. Azzini, A., Tettamanzi, A.: Evolving neural networks for static single-position. Journal of Artificial Evolution and Applications 2008, 1–17 (2008) Article ID 184286
15. Barber, B., Odean, T.: Trading is hazardous to your wealth: The common stock investment performance of individual investors. Journal of Finance 55, 773–806 (2000)
16. Brabazon, A., O'Neill, M. (eds.): Natural computing in computational economics and finance. Springer, Heidelberg (2008)
17. Brenner, T.: Agent learning representation: Advice on modeling economic learning. In: Tesfatsion, L., Judd, K. (eds.) Handbook of computational economics: Agent-based computational economics, vol. 2, pp. 895–947. Elsevier, Oxford (2006)
18. Bonissone, P.: Hybrid soft computing systems: Where are we going? In: Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000), Berlin, Germany, pp. 739–746 (2000)
19. Carlin, B., Louis, T.: Bayes and empirical Bayes methods for data analysis, 2nd edn. CRC Press, Boca Raton (2000)
20. Casella, G.: An introduction to empirical Bayes data analysis. The American Statistician 39, 83–87 (1985)
21. Chan, N., LeBaron, B., Lo, A., Poggio, T.: Information dissemination and aggregation in asset markets with simple intelligent traders, Working paper. MIT, Cambridge (1999)
22. Chen, S.-H.: Software-agent designs in economics: An interdisciplinary framework. IEEE Computational Intelligence Magazine 3(4), 22–26 (2008a)
23. Chen, S.-H.: Computational intelligence in agent-based computational economics. In: Fulcher, J., Jain, L. (eds.) Computational intelligence: A compendium, pp. 517–594. Springer, Heidelberg (2008)
24. Chen, S.-H., Kuo, T.-W.: Overfitting or poor learning: A critique of current financial applications of GP. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 34–46. Springer, Heidelberg (2003)
25. Chen, S.-H., Huang, Y.-C.: Risk preference, forecasting accuracy and survival dynamics: Simulations based on a multi-asset agent-based artificial stock market. Journal of Economic Behavior and Organization (forthcoming) (2008)

26. Chen, S.-H., Tai, C.-C.: Trading restrictions, price dynamics, and allocative efficiency in double auction markets: Analysis based on agent-based modeling and simulations. Advances in Complex Systems 6(3), 283–302 (2003)
27. Chen, S.-H., Tai, C.-C.: On the selection of adaptive algorithms in ABM: A computational-equivalence approach. Computational Economics 28(1), 51–69 (2006)
28. Chen, S.-H., Tai, C.-C.: Would human agents like software agents? Results from prediction market experiments. Working paper, AI-ECON Research Center, National Chengchi University (2007)
29. Chen, S.-H., Wang, P.: Computational intelligence in economics and finance. Springer, Heidelberg (2003)
30. Chen, S.-H., Yeh, C.-H.: Genetic programming learning and the cobweb model. In: Angeline, P. (ed.) Advances in Genetic Programming, vol. 2, ch. 22, pp. 443–466. MIT Press, Cambridge (1996)
31. Chen, S.-H., Yeh, C.-H.: Modeling speculators with genetic programming. In: Angeline, P.J., McDonnell, J.R., Reynolds, R.G., Eberhart, R. (eds.) EP 1997. LNCS, vol. 1213, pp. 137–147. Springer, Heidelberg (1997)
32. Chen, S.-H., Yeh, C.-H.: Simulating economic transition processes by genetic programming. Annals of Operation Research 97, 265–286 (2000)
33. Chen, S.-H., Yeh, C.-H.: Evolving traders and the business school with genetic programming: A new architecture of the agent-based artificial stock market. Journal of Economic Dynamics and Control 25, 363–394 (2001); Chen, S.-H., Kuo, T.-W., Shieh, Y.-P.: Genetic Programming: A Tutorial with the software Simple GP. In: Chen, S.-H. (ed.) Genetic algorithms and genetic programming in computational finance, pp. 55–77. Kluwer, Dordrecht (2002)
34. Chen, S.-H., Liao, C.-C., Chou, P.-J.: On the plausibility of sunspot equilibria: Simulations based on agent-based artificial stock markets. Journal of Economic Interaction and Coordination 3(1), 25–41 (2008)
35. Chen, S.-H., Wang, P., Kuo, T.-W.: Computational intelligence in economics and finance, vol. 2. Springer, Heidelberg (2007)
36. Chen, S.-H., Zeng, R.-J., Yu, T.: Co-evolving trading strategies to analyze bounded rationality in double auction markets. In: Riolo, R. (ed.) Genetic programming theory and practice VI. Springer, Heidelberg (forthcoming) (2008)
37. Chen, W.-H., Shih, J.-Y., Wu, S.: Comparison of support-vector machines and back propagation neural networks in forecasting the six major Asian stock markets. International Journal of Electronic Finance 1(1), 49–67 (2006)
38. Chen, P., Quek, C., Mah, M.: Predicting the impact of anticipatory action on U.S. stock market– An event study using ANFIS (a neural fuzzy model). Computational Intelligence 23(2), 117–141 (2007)
39. Chokri, S.: Neuro-fuzzy network based on extended Kalman filtering for financial time series. In: Proceedings of World Academy of Science, Engineering and Technology, vol. 15, pp. 290–295 (2006)
40. Choudhry, R., Garg, K.: A hybrid machine learning system for stock market forecasting. In: Proceedings of the World Academy of Science, Engineering and Technology, vol. 29, pp. 315–318 (2008)
41. Cordon, O., Herrera, F., Hoffmann, F., Magdalena, L.: Genetic fuzzy systems: Evolutionary tuning and learning of fuzzy knowledge bases. World Scientific, Singapore (2002)
42. De Grauwe, P., Grimaldi, M.: The exchange rate in a behavior finance framework. Princeton University Press, Princeton (2006)

43. de Jong, E., Verschoor, W., Zwinkels, R.: Heterogeneity of agents and exchange rate dynamics: Evidence from the EMS, `http://ssrn.com/abstract=890500`
44. Delli Gatti, D., Gaffeo, E., Gallegati, M., Giulioni, G.: Emergent macroeconomics: An agent-based approach to business fluctuations. Springer, Heidelberg (2008)
45. Duffy, J.: Agent-based models and human subject experiments. In: Tesfatsion, L., Judd, K. (eds.) Handbook of computational economics: Agent-based computational economics, vol. 2, pp. 949–1011. Elsevier, Oxford (2006)
46. Duval, Y., Kastens, T., Featherstone, A.: Financial classification of farm businesses using fuzzy systems. In: 2002 AAEA Meetings Long Beach, California (2002)
47. Evans, G., Honkapohja, S.: Learning and expectations in macroeconomics. Princeton University Press, Princeton (2001)
48. Feinberg, M.: Why smart people do dumb things: Lessons from the new science of behavioral economics, Fireside (1995)
49. Fogel, D., Chellapilla, K., Angeline, P.: Evolutionary computation and economic models: sensitivity and unintended consequences. In: Chen, S.-H. (ed.) Evolutionary computation in economics and finance, pp. 245–269. Springer, Heidelberg (2002)
50. Frankel, J., Froot, K.: Chartists, fundamentalists, and trading in the foreign exchange market. American Economic Review 80, 181–186 (1990)
51. Gilks, W., Richardson, S., Spiegelhalter, D.: Markov chain Monte Carlo in practice. CRC, Boca Raton (1995)
52. Gode, D., Sunder, S.: Allocative efficiency of markets with zero intelligence traders: Market as a partial substitute for individual rationality. Journal of Political Economy 101, 119–137 (1993)
53. Grossklags, J., Schmidt, C.: Software agents and market (in)efficiency: a human trader experiment. IEEE Transactions on Systems, Man, and Cybernetics, Part C 36(1), 56–67 (2006)
54. Hartley, J.: The representative agent in macroeconomics. Routledge, London (1997)
55. Henrich, J.: Does culture Matter in Economic Behavior? Ultimatum game bargaining among the Machiguenga of the Peruvian Amazon. American Economic Review 90(4), 973–979 (2000)
56. Henrich, J., Boyd, R., Bowles, S., Camerer, C., Fehr, E., Gintis, H. (eds.): Foundations of human sociality: Economic experiments and ethnographic evidence from fifteen small-scale societies. Oxford University Press, Oxford (2004)
57. Herrnstein, R., Murray, C.: Bell curve: Intelligence and class structure in American life. Free Press (1996)
58. Hommes, C.: Heterogeneous agent models in economics and finance. In: Tesfatsion, L., Kenneth, J. (eds.) Handbook of Computational Economics, vol. 2, ch. 23, pp. 1109–1186. Elsevier, Amsterdam (2006)
59. Jang, R.: ANFIS: Adaptive network-based fuzzy inference system. IEEE Transactions on Systems, Man and Cybernetics 23(3), 665–685 (1993)
60. Janssen, M., Ostrom, E.: Empirically based, agent-based models. Ecology and Society 11(2), 37 (2006)
61. Jensen, A.: The g Factor: The science of mental ability, Praeger (1998)
62. Kakati, M.: Option pricing using the adaptive neuro-fuzzy system (ANFIS). ICFAI Journal of Derivatives Markets 5(2), 53–62 (2008)
63. Kendall, G., Yao, X., Chong, S.-Y.: The iterated prisoners' dilemma: 20 years on. World Scientific, Singapore (2007)
64. Kim, K.-J.: Artificial neural networks with evolutionary instance selection for financial forecasting. Expert Systems with Applications 30(3), 519–526 (2006)

65. Koyama, Y., Sato, H., Matsui, H., Nakajima, Y.: Report on UMIE 2004 and summary of U-Mart experiments based on the classification of submitted machine agents. In: Terano, T., Kita, H., Kaneda, T., Arai, K., Deguchi, H. (eds.) Agent-based simulation: From modeling methodologies to real-world applications. Springer Series on Agent-Based Social Systems, vol. 1, pp. 158–166 (2005)
66. Kosko, B.: Neural networks and fuzzy systems. Prentice-Hall, Englewood Cliffs (1992)
67. LeBaron, B.: Evolution and time horizons in an agent based stock market. Macroeconomic Dynamics 5, 225–254 (2001)
68. Lieberman, H.: Software agents: The MIT approach. In: Invited speech delivered at the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW 1996), Eindhoven, The Netherlands, January 22-25 (1996)
69. Lubinski, D., Humphreys, L.: Incorporating general intelligence into epidemiology and the social sciences. Intelligence 24(1), 159–201 (1997)
70. Lucas Jr., R.: Adaptive behavior and economic theory. Journal of Business 59, 401–426 (1986)
71. Lynn, R.: Race differences in intelligence: An evolutionary analysis. Washington Summit Publishers (2006)
72. Lynn, R., Vanhanen, T.: IQ and the wealth of nations. Praeger (2002)
73. Malhotra, R., Malhotra, D.: Differentiating between good credits and bad credits using neuro-fuzzy systems. European Journal of Operational Research 136(1), 190–211 (2002)
74. Markose, S.: Developments in experimental and agent-based computational economics (ACE): overview. Journal of Economic Interaction and Coordination 1(2), 119–127 (2006)
75. Manzan, S., Westerhoff, F.: Heterogeneous expectations, exchange rate dynamics and predictability. Journal of Economic Behavior and Organization 64, 111–128 (2007)
76. McClearn, G., Johansson, B., Berg, S., Pedersen, N., Ahern, F., Petrill, S., Plomin, R.: Substantial genetic influence on cognitive abilities in twins 80 or more years old. Science 276, 1560–1563 (1997)
77. McKee, T., Lensberg, T.: Genetic programming and rough sets: A hybrid approach to bankruptcy classification. European Journal of Operational Research 138(2), 436–451 (2002)
78. Mora, A., Alfaro-Cid, E., Castillo, P., Merelo, J., Esparcia-Alcazar, A., Sharman, K.: Discovering causes of financial distress by combining evolutionary algorithms and artificial neural networks. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008), Atlanta, USA (July 2008)
79. Murray, C.: Income Inequality and IQ. AEI Press (1998)
80. Nauck, D., Klawonn, F., Kruse, R.: Foundations of neuro-fuzzy systems. John Wiley and Sons, Chichester (1997)
81. Pai, P.-F.: System reliability forecasting by support vector machines with genetic algorithms. Mathematical and Computer Modelling 43(3-4), 262–274 (2006)
82. Palmer, R., Arthur, B., Holland, J., LeBaron, B., Tayler, P.: Artificial economic life: A simple model of a stock market. Physica D 75, 264–274 (1994)
83. Phua, P., Ming, D., Li, W.: Neural network with genetically evolved algorithms for stocks prediction. Asia-Pacific Journal of Operational Research 18(1), 103–107 (2001)
84. Plomin, R., Petrill, S.: Genetics and intelligence: What's new? Intelligence 24(1), 53–77 (1997)
85. Richiardi, M., Leombruni, R., Contini, B.: Exploring a new ExpAce: The complementarities between experimental economics and agent-based computational economics. Journal of Social Complexity 3(1) (2006)

86. Rust, J., Miller, J., Palmer, R.: Behavior of trading automata in a computerized double auction market. In: Friedman, D., Rust, J. (eds.) The double auction market: Institutions, theories, and evidence, ch. 6, pp. 155–198. Addison Wesley, Reading (1993)
87. Rust, J., Miller, J., Palmer, R.: Characterizing effective trading strategies: Insights from a computerized double auction market. Journal of Economic Dynamics and Control 18, 61–96 (1994)
88. Salcedo-Sanz, S., Fernandez-Villacanas, J.-L., Segovia-Vargas, M., Bousono-Calzon, C.: Genetic programming for the prediction of insolvency in non-life insurance companies. Computers & Operations Research 32(4), 749–765 (2005)
89. Sargent, T.: Bounded rationality in macroeconomics. Oxford University Press, Oxford (1993)
90. Sasaki, Y., Flann, N., Box, P.: Multi-agent evolutionary game dyanmics and reinforcement learning applied to online optimization for the traffic policy. In: Chen, S.-H., Jain, L., Tai, C.-C. (eds.) Computational economics: A perspective from computational intelligence. IDEA Group Publishing, USA (2005)
91. Sato, H., Matsui, H., Ono, I., Kita, H., Terano, T., Deguchi, H., Shiozawa, Y.: U-Mart project: Learning economic principles from the bottom by both human and software agents. In: Terano, T., Nishida, T., Namatame, A., Tsumoto, S., Ohsawa, Y., Washio, T. (eds.) JSAI-WS 2001. LNCS, vol. 2253, pp. 121–131. Springer, Heidelberg (2001)
92. Sato, H., Kawachi, S., Namatame, A.: The statistical properties of price fluctuations by computer agents in a U-Mart virtual future market simulator. In: Terano, T., Deguchi, H., Takadama, K. (eds.) Meeting the challenge of social problems via agent-based simulation, pp. 67–76. Springer, Heidelberg (2003)
93. Selvaratnam, S., Kirley, M.: Predicting stock market time series using evolutionary artificial neural networks with Hurst exponent windows. In: Sattar, A., Kang, B.-h. (eds.) AI 2006. LNCS, vol. 4304, pp. 617–626. Springer, Heidelberg (2006)
94. Shahwan, T., Odening, M.: Forecasting agricultural commodity prices using hybrid neural networks. In: Chen, S.-H., Wang, P., Kuo, T.-W. (eds.) Computational intelligence in economics and finance, vol. 2, pp. 63–74. Springer, Heidelberg (2007)
95. Shiozawa, Y., Nakajima, Y., Matsui, H., Koyama, Y., Taniguchi, K., Hashimoto, F.: Artificial Market Experiments with the U-Mart. Springer, Tokyo (2006)
96. Smith, V.: Bidding and auctioning institutions: Experimental results. In: Smith, V. (ed.) Papers in eExperimental economics, pp. 106–127. Cambridge University Press, Cambridge (1991)
97. Su, H., Yu, S.: Hybrid GA based online support vector machine model for short-term traffic flow forecasting. In: Xu, M., Zhan, Y.-W., Cao, J., Liu, Y. (eds.) APPT 2007. LNCS, vol. 4847, pp. 743–752. Springer, Heidelberg (2007)
98. Tay, N., Linn, S.: Fuzzy inductive reasoning, expectation formation and the behavior of security prices. Journal of Economic Dynamics & Control 25, 321–361 (2001)
99. Terano, T., Shiozawa, Y., Deguchi, H., Kita, H., Matsui, H., Sato, H., Ono, I., Kakajima, Y.: U-Mart: An artificial market testbed for economics and multiagent systems. In: Terano, T., Deguchi, H., Takadama, K. (eds.) Meeting the challenge of social problems via agent-based simulation, pp. 53–66. Springer, Heidelberg (2003)
100. Tung, W.-L., Quek, C., Cheng, P.: GenSo-EWS: A novel neural-fuzzy based early warning system for predicting bank failures. Neural Networks 17(4), 567–587 (2003)
101. Ueda, T., Taniguchi, K., Nakajima, Y.: An analysis of U-Mart experiments by machine and human agents. In: Proceedings of 2003 IEEE international symposium on computational intelligence in robotics and automation, vol. 3, pp. 1340–1347 (2003)
102. Wang, S.-C., Li, S.-P., Tai, C.-C., Chen, S.-H.: Statistical properties of an experimental political futures markets. Quantitative Finance (2007) (forthcoming)

103. Wu, C.H., Tseng, G.H., Goo, Y.J., Fang, W.C.: A real-valued genetic algorithm to optimize the parameters of support vector machine for Predicting bankruptcy. Expert Systems with Applications 32(2), 397–408
104. Xiong, Z.-B., Li, R.-J.: Credit risk evaluation with fuzzy neural networks on listed corporations of China. In: Proceedings of 2005 IEEE International Workshop on VLSI Design and Video Technology, pp. 397–402 (2005)
105. Yu, L., Zhang, Y.-Q.: Evolutionary fuzzy neural networks for hybrid financial prediction. IEEE Transactions on Systems, Man and Cybernetics, Part C 35(2), 244–249 (2005)
106. Yu, L., Lai, K.-K., Wang, S.: An evolutionary programming based SVM ensemble model for corporate failure prediction. In: Beliczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B. (eds.) ICANNGA 2007. LNCS, vol. 4432, pp. 262–270. Springer, Heidelberg (2007)
107. Yu, L., Wang, S.-Y., Lai, K.K.: Mining stock market tendency using GA-based support vector machines. In: Deng, X., Ye, Y. (eds.) WINE 2005. LNCS, vol. 3828, pp. 336–345. Springer, Heidelberg (2005)
108. Yunos, Z., Shamsuddin, S., Sallehuddin, R.: Data modeling for Kuala Lumpur Composite Index with ANFIS. In: Proceedings of 2008 Second Asia International Conference on Modelling & Simulation, pp. 609–614 (2008)
109. Zopounidis, C., Doumpos, M., Pardalos, P. (eds.): Handbook of financial engineering. Springer, Heidelberg (2008)

IMMUNE: A Collaborating Environment for Complex System Design

Mahmoud Efatmaneshnik and Carl Reidsema

Abstract. Engineering complex systems is a testing paradigm for engineers of this century. Integration of complex systems design is accomplished through innovation, and autonomy of design agents has been recognized as the main contributor to novelty and innovation of solutions. Unfortunately, agents' autonomy can make the design environment chaotic and inefficient. To address this dilemma of distributed versus central control in complex system design, decision support systems that enable robust collaboration amongst many design agents from different disciplines, are required. The particular characteristics of such decision support system must include immunity to catastrophic failures and sudden collapse that are usually observed in complex systems. This chapter lays the conceptual framework for IMMUNE as a robust collaborating design environment. In this environment the complexity arising from autonomous collaborations is sensed and monitored by a central unit. The collaboration complexity, which is the collective problem solving capability of the design system, is compared to the complexity of the problem estimated from simulation based techniques. In this regard IMMUNE is an artificial immune system that balances the complexity of the environment and by that increases the possibility of achieving innovative and integral solutions to the complex design problems. Agents in IMMUNE are adaptive and can change their negotiation strategy and by that can contribute to the overall capability of the design system to maintain its problem solving complexity.

Mahmoud Efatmaneshnik
Design Research Laboratory, School of Mechanical and Manufacturing Engineering,
The University of New South Wales, Sydney NSW 2052, Australia
e-mail: mahmoud@student.unsw.edu.au

Carl Reidsema
Design Research Laboratory, School of Mechanical and Manufacturing Engineering,
The University of New South Wales, Sydney NSW 2052, Australia
e-mail: reidsema@unsw.edu.au

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 275–320.
springerlink.com

1 Introduction

A decision is a choice between alternatives based on estimates of the values of those alternatives. There is a substantial amount of empirical evidence that human intuitive judgment and decision making can be far from optimal, and it deteriorates even further with the complexity of the problem and stress [21]. Supporting a decision means helping people working alone or in a group to gather intelligence, generate alternatives and make choices. A Decision Support System (DSS) is a computerized system for helping make decisions. Supporting the decision process involves supporting the estimation, the evaluation and/or the comparison of alternatives (see Turban [68]). Turban defines a DSS more specifically as "an interactive, flexible, and adaptable computer-based information system, especially developed for supporting the solution of a non-structured management problem for improved decision making". Druzdzel and Flynn [21] define a DSS as an integrated computing environment for complex decision making. A DSS can be defined as a knowledge-based system, which formalizes the domain knowledge so that it is amenable to mechanized reasoning. Knowledge-based problem solving is the domain of Artificial Intelligence (AI) and the selection of an appropriate AI development tool that may provide a framework to incorporate knowledge will come from this area (see Reidsema and Szczerbicki [54]). Reidsema and Szczerbicki identified three different architectures for decision support systems for product design planning and manufacturing in a concurrent engineering environment: Expert Systems, Agent Based Systems, and Blackboard Database Systems. These have been defined as follows:

An *Expert system* is one of a class of AI techniques that is able to capture the knowledge and reasoning of an experienced expert for re-use in assisting the less experienced in making decisions.

The *Blackboard Database Architecture* is a problem solving system based on the metaphor of human experts who cooperate by entering partial solutions to the current problem onto a physical blackboard. The type of problems best suited to this approach are those that are able to be reduced to a set of simpler problems that are reasonably independent. The integration of the partial solutions to the overall solution takes place by the intervention of a centralized controller known as control source and therefore has a top down approach to problem solving.

Multi-agent systems are distributed systems that use a bottom up approach to problem solving in which case the intervention of the centralized coordination between agents is minimal or totally eliminated. Each agent in a multi-agent system behaves as an abstraction tool which has the characteristics of a self-contained problem solving system that is capable of autonomous, reactive, proactive as well as interactive behaviour. The solution in this case emerges as a whole and is the result of a synergetic effect. Synergy denotes a level of group performance that is above and beyond which could be achieved by the members of the group working independently [40]. Synergy in a multi-agent system enables the integration of partial solutions of nonlinear and coupled problem. Multi-agent systemsindexmulti-agent system are the natural candidates for complex systems which show heavy interdependency between partial problems.

Providing an extensive literature review of concurrent design and manufacturing systems, Shen et al. [59] identify three different approaches for agent based architectures: hierarchical architectures, federated architectures and autonomous agent architectures. Each architecture has particular strengths for specific applications, and choosing the right architecture involves matching requirements to capabilities. Hierarchical architectures consist of semi-autonomous agents with a global control agent dictating goals/plans or actions to the other agents. Multi-Agent Systems with a global blackboard data base are hierarchical architectures. They report that some researchers have considered their blackboard systems to be multi-agent systems, and others have implemented their agent based systems using blackboard architectures. In these systems control can be implemented in different ways: using a special control expert called a supervisor as in EXPORT [48]; using a shared graphical model as in ICM, “Interdisciplinary Communication Medium” [26] or a shared database as in SHARED [70]; or through multiple shared workspaces as in MATE [56]. Because hierarchical architectures suffer from deficiencies associated with their centralized character, federated multi-agent architectures are increasingly being considered as a compromise solution for industrial agent based applications [59], especially for large scale engineering applications. A fully federated agent based system has no explicit shared facility for storing active data; rather, the system stores all data in local databases and handles updates and changes through message passing. In theory, a truly open multi-agent system need not have any predefined global control. An example of such an architecture is DIDE, “Distributed Intelligent Design Environment” [57]. Another good example of such a system is ANARCHY which was a working prototype of an asynchronous design environment [50]. Agents in ANARCHY were autonomous, and used broadcast communications. It however, utilised a global design strategy based on simulated annealing. For such systems there is the threat of exhibiting chaotic behaviour [65].

In this paper we present IMMUNE which is a flat federated architecture for the parametric design of complex products. IMMUNE uses a global blackboard to save the current state of the design. All the agents are grouped into virtual teams or coalitions, the structure of which mirrors the structure of the problem and its decomposition pattern. This idea was previously utilized in MetaMorph [47]. MetaMorph was devised as an adaptive agent based architecture to address system adaptation and extended-enterprise issues at four fundamental levels: virtual enterprise, distributed intelligent systems, concurrent engineering, and agent architecture. MetaMorph benefited extensively from low level mediators to coordinate between different groups (or coalitions). IMMUNE, however, does not have any low level mediator, broker or facilitator, and is a flat architecture. Instead IMMUNE benefits from a special unit in the control shell of the blackboard that we denote as the CEO (Complexity Estimator and Observer). The CEO estimates the complexity measure of the problem and compares it to the observed complexity of the multi-agent system. In IMMUNE the problems at various levels of abstraction are decomposed using a complexity measure of the problem [24]. Based on the complexity of the problem after decomposition (real complexity), the CEO estimates the minimum and maximum complexity of the process. It then monitors the complexity of the process as a function of the exchanged information between the agents (cognitive complexity). An *effective* and *efficient* design process must have a cognitive complexity in between the minimum and maximum complexity of the problem; this is the result of a simple notion which is: the best a single person (or a single system) can do is limited by his/her (cognitive) complexity [2].

A design system, with its cognitive complexity surpassing the maximum complexity of the problem, has lost *effectiveness* since the design process may become chaotic. If the cognitive complexity of the design system is lower than the minimum complexity of the problem, then the *efficiency* of the system, in solving the complex problem and managing the interdependencies between its subproblems, would not be achieved. In both cases, the agents are expected to undertake corrective measures to stabilize the cognitive complexity of the system and immunize it against fragility, and failure. The CEO monitors the complexity at two levels: inside the coalitions (at the local levels) and the entire system (at the federal level). The next section discusses the fundamentals of design planning for complex products and a complexity based method for monitoring the design process.

2 Design Planning for Complex Products Development

Real world concurrent engineering design projects require the cooperation of multidisciplinary design teams using sophisticated and powerful engineering tools such as commercial CAD tools, engineering database systems and knowledge based systems [51]. To coordinate the design activities of various groups and support effective cooperation among the different engineering tools, a distributed intelligent environment is required. This environment should not only automate individual tasks, in the manner of traditional computer aided engineering tools, but also help individual members share information and coordinate their actions as they explore alternatives in search of a globally optimal or near optimal solution. Designing in a concurrent environment requires the precise planning of the resources, tasks, collaborations, information exchanges and cooperation [53]. Planning to achieve the development of a new product is usually accomplished by distributing the tasks required to achieve the plan to individuals or groups of team members, best suited to accomplishing these tasks [53]. Planning increases the design efficiency, and reduces the risk of not achieving a design consensus and consequently the agreed upon design objectives.

There are various approaches and perspectives to design planning which Reidsema and Szczerbicki [53] summarized as:

- Task model
- Design Process
- Resource Structure
- Organizational Model
- Cooperative Planning Model (GDDI)

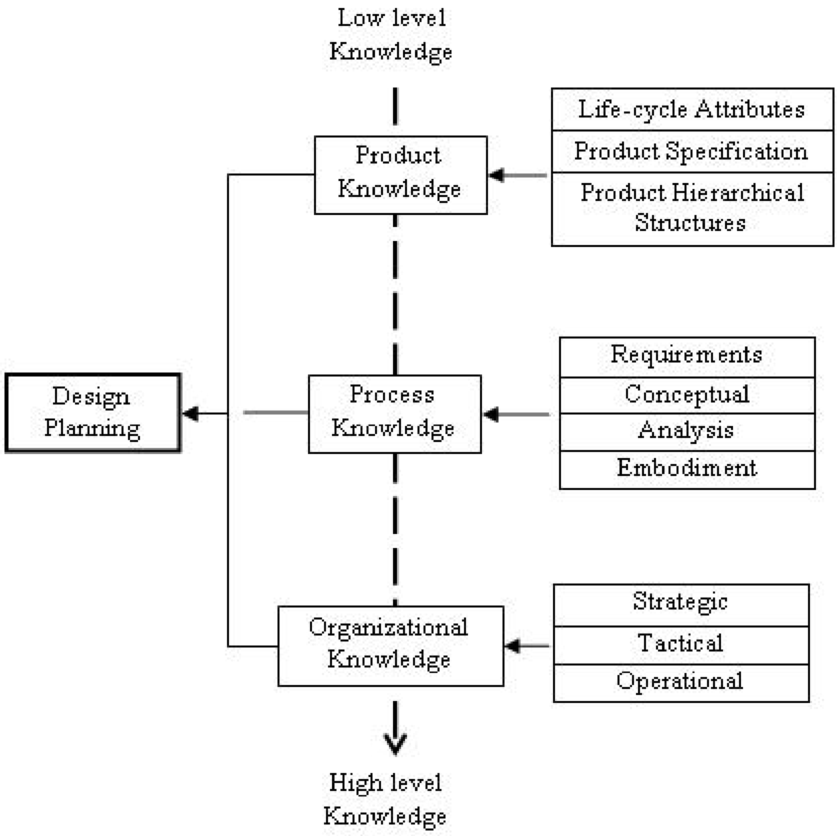


Fig. 1 Design planning can be performed by using knowledge corresponding to various levels of the hierarchy of organization, process and product [53]

The cooperative model of design planning proposed by Reidsema and Szczerbicki [52] deploys a gradualistic approach to dealing with problems of a complex nature (such as planning in the concurrent engineering environment), through the gradual introduction of the problem space. The full cycle of distributed planning may then be thought to consist of:

1. Plan Generation,
2. Plan Decomposition
3. Plan Distribution
4. Plan Integration

The execution of this cycle of Generation, Decomposition, Distribution, and Integration (GDDI cycle) follows a common planning method whereby a complex problem is first reduced to a number of simpler problems. These simpler problems are then solved and the information obtained from their solution is used to solve the parent problem. The coordination and control of these agents is achieved through

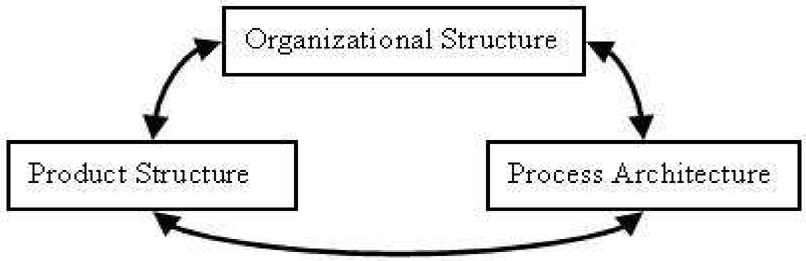


Fig. 2 Produce, process and organization structures are tightly related [7]

the multi-agent planning actions of task decomposition, task distribution and result integration [53].

For complex systems, due to coupling between the distributed tasks, integration may not be performed linearly simply by adding the partial solutions together. Since the coupled problems tend to be nonlinear (same as the coupled differential equations) as a result the solutions may not be achieved by using the usual concurrent planning (that adds the partial solutions to obtain the overall solution). The nonlinearity limits the kind of knowledge being used for planning. In general in a CE environment, planning may be obtained by the utilization of quite diverse knowledge from the Product, Process and Organizational (PPO) knowledge domains (Figure 1).

2.1 The Necessity of Emulating Low Level Product Knowledge for Complex Product Design Planning

The design structure matrix (DSM) is a well known knowledge representation and analysis tool for system modeling, especially for the purpose of decomposition and integration. A DSM displays the relationships between components of a system in a compact, visual, and analytically advantageous format as a square matrix with identical row and column labels. DSMs are usually employed in modeling products, processes, and organizational architectures. Presenting the following definitions, Browning [7] argued that the three DSMs and the structures they model are tightly related, and in many real industrial cases they exhibit strong couplings (Figure 2):

1. *Parameter-Based (or Low-Level Schedule) DSM:* Used for modeling low-level relationships between design decisions and parameters, systems of equations, subroutine parameter exchanges which represents the product architecture.
2. *Activity-Based or Schedule DSM:* Used for modeling processes and activity networks based on activities and their information flow and other dependencies.
3. *Team-Based or Organization DSM:* Used for modeling organization structures based on people and/or groups and their interactions.

Most design planning takes place in a top-down fashion by defining and decomposing the organization structure. If they begin at "the top", such models rarely reach the lowest levels of design activity, where individual design parameters are determined based on other parameters. Determining these parameters constitutes the lowest level design activities, and a bottom-up, integrative analysis of these low-level activities can provide process structure insights. Browning [7] emphasized that clearly, parameter-based DSMs have integrative applications. This characteristic of the parameter based DSM which represents the low level product knowledge makes it suitable to be utilized in the planning of complex engineering systems. We have previously referred to this matrix as the self of the system [23] with the values of parameters representing the non self.

2.2 Concurrent Parametric Design

Emulation of low level product knowledge for planning is feasible through simplification of the design problem, by modeling the target product as sets of variables. These include the set of design variables (or inputs) and the set of design objectives (or outputs). Within the context of planning, the outcome of the decision making process is quite likely to be an objective [53]. Design objectives are concerned with the performance and quality of the product, lead time and cost of production. The design variable sets include subsets of sizing variables, shape variables, topologies and process knowledge and manufacturing variables such as process capabilities (see Prasad [49]). Prasad has defined some of these as follows:

- *Sizing Variables*: these include variables like thicknesses (for thin walled sections) and areas (for solid objects) that can be changed.
- *Shape Variables*: These involve changing the configuration points or the geometry of the parts that are represented such as length width, height, coordinates and so on.
- *Topology Variables*: These define parameters that actually determine where material should or should not be removed. As long as the topology change can be represented parametrically in the CAD system, the model can be optimized. Topology optimization allows feature patterns such as how many bolts are needed to hold down a given part, or how many ribs provide a given stiffness.
- *Process Variables*: these involve changing the rules concerning the part's forming or processing needs that have the effect on changing the part's size, shape, topology or functions themselves, cost and lead time.
- *Manufacturing Variables*: these include the process capability indices, and required precisions, manufacturing lead time and cost.

Each variable may be accompanied by a set of constraints. Design problem solving is the process of assigning values to parameters in accordance with the given design requirements, constraints, and optimization criterion [71]. A design task in this view constitutes the determination of a single design variable and a plan comprises a set of design parameters with their constraints, and determining the values of which is the task of multidisciplinary design teams.

Chen and Li [9] referred to Concurrent Product Design taking place in the parametric design stage as Concurrent Parametric Design. An engineered product is developed through the concurrent consideration of various design issues, and multiple teams may be needed to tackle different design issues. In parametric design, the focus is on the determination of a parametric configuration that achieves an optimization of individual design attributes. In multi-team design, a team refers to a collaboration of design participants that, in a broad sense, can consist of designers, computers or even algorithms, whatsoever is able to cope with distributed tasks as part of the whole design problem [9]. In this design situation, teams may face uncertainties during the design process, especially when their design decisions are interrelated [9].

2.3 Simulation Based Engineering, and Complexity Measures

Formica [25] has recognized *modeling and simulation* as the key to reconcile ambitious performance and operational requirements improvement with realistic development and production costs, times and risks for highly innovative industrial high-tech systems. The performance and operational requirements are micro or low level parameters whereas production costs, times and risks are macro and high level (often emergent), properties; a system exhibits emergence when there are coherent properties at the macro-level (i.e. of the system as a whole) that dynamically arise from the interactions between the parts at the micro-level [69]. Integrating these two levels of micro and macro level knowledge in a Cyber Infrastructural Environment, Formica [25] addresses the issue of engineering process fragmentation. Connecting the low (micro) level design variables and performance variables to high (macro) level organizational goals including cost and lead time is referred to as “Knowledge Integrators and Multipliers” because they are able to synthesize data, information, and knowledge from [25]:

- Different disciplines.
- A wide range of scientific and engineering time and space domains.
- Multiple scientific and engineering models and representations (science-engineering integration).
- Multiple methods (analytical theories and experimentation & testing) and related knowledge bases.

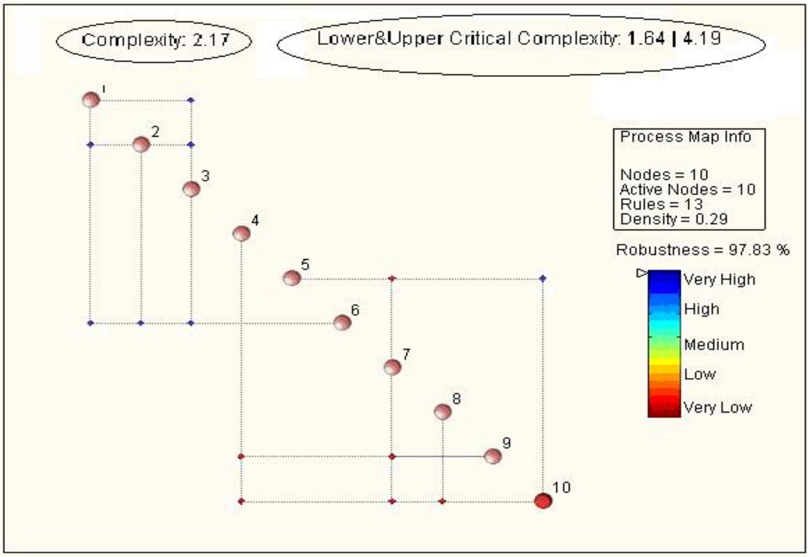
As creating high-fidelity simulation models is a complex activity that can be quite time-consuming [60], Monte Carlo Simulation is suggested to establish the fitness landscape of the design problem. A fitness landscape is a multi-dimensional data set, in which the number of dimensions is determined by the number of system variables [46]. Marczyk [45] has stressed that by means of Monte Carlo Simulation of design parameters (at both micro and macro levels), the fitness landscape of the design space is created, enabling the verification of the global dependencies between

Table 1 A simulated DSM is a weighted adjacency matrix. This DSM has 10 variables

-	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	0	0.53	0.32	0	0	0.1	0	0	0	0
V2	0.76	0	0.12	0.12	0.3	0.2	0.2	0	0.1	0
V3	0.45	0.11	0	0	0	0.2	0.3	0	0.52	0.72
V4	0.16	0.65	0.64	0	0.34	0.43	0	0	0	0
V5	0.22	0.44	0.11	0.45	0	0.53	0.02	0	0.02	0
V6	0.77	0.78	0.31	0.34	0	0	0	0	0	0
V7	0.12	0	0.02	0	0	0	0	0.45	0.1	0.3
V8	0.01	0	0	0	0.01	0	0.2	0	0.4	0.1
V9	0	0	0.15	0	0	0	0.7	0.2	0	0.5
V10	0	0.18	0	0	0.01	0	0.1	0.8	0.9	0

low level design variables (product characteristics) and high level design process variables (cost, time). We suggest the actualization of the multi scale and multidisciplinary design structure matrices through Monte Carlo and Statistical Simulation in the early stage of the design process, after the plan generation phase in each GDDI cycle and before the plan decomposition and distribution. In order to establish the correlation coefficients between different variables, global entropy based correlation coefficient have significant advantages over linear covariance analyses through capturing both linear and nonlinear dependencies. This measure is embedded in the Ontospace^{TM} software. The outcome of the design space (or fitness landscape) simulation may be fed into this software to produce the correlation matrix (simulated parameter based DSM). Table 1 shows an example of a typical simulated parameter based DSM of a product (such as an aircraft, car, computes, etc) with normalized weights (all the weights are between zero and one). The data presented and the design variables are however fictitious. Throughout this paper, the self map will be referred to as the corresponding weighted graph of the matrix in Table 1.

Complexity is frequently confused with emergence; emergence of new structures and forms is the result of re-combination and spontaneous self-organization of simpler systems to form higher order hierarchies, i.e. a result of complexity [46]. We define complexity as the intensity of emergence in a system. If the complexity is too high the system becomes chaotic and uncontrollable and is likely to lose its structure, or in other words, downward causation raises the subsystems' performance. If complexity is too low the system loses the intrinsic characteristics of the entity it was intended to describe, and fails to emerge as a spontaneous organization. Complexity materializes the system's self by the emergence of the self structure when the subsystems have sufficient interaction. Complexity is a "holistic" measure of the system that enables us to study the system as a "whole". Marczyk and Deshpande [46] proposed a comprehensive complexity metric and established a conceptual platform for practical and effective complexity management. The metric takes into account all significant aspects necessary for a sound and comprehensive complexity measure,



Complexity: 2.17

Lower&Upper Critical Complexity: 1.64 | 4.19

Process Map Info

Nodes = 10
Active Nodes = 10
Rules = 13
Density = 0.29

Robustness = 97.83 %

Fig. 3 An example of a self map and its three complexity measures

namely structure, entropy and data granularity, or coarse-graining [46]. The metric allows one to relate complexity to fragility and to show how critical threshold complexity levels may be established for a given system. The metric is incorporated into OntoSpace^{TM}, an innovative complexity management software. This software calculates three complexity measures for every self map (Figure 3):

1. The complexity of the map which is a very specific measure reflecting the coupled-ness and size of the system. This complexity measure is called Ontix. We will refer to the complexity of this map as self complexity.
2. The upper complexity bound to which the complexity of the system may be increased without exhibiting chaos.
3. The lower complexity bound where the system with a lower complexity loses its intrinsic characteristics and fails to emerge as a coherent self.

"A system performing a given basic function is irreducibly complex if it includes a set of well-matched, mutually interacting, nonarbitrarily individuated parts such that each part in the set is indispensable to maintaining the system's basic, and therefore original, function. The set of these indispensable parts is known as the irreducible core of the system" [17]. The lower complexity bound represents the irreducible complexity of the system that contains the intrinsic characteristics of the system.

There is a sufficient body of knowledge to sustain the belief that whenever dynamical systems undergo a catastrophe, the event is accompanied by a sudden jump in complexity [46]; this is also intuitive: a catastrophe implies loss of functionality, or organization. The increase of entropy increases complexity – entropy is not necessarily adverse as it can help to increase fitness – but at a certain point, complexity

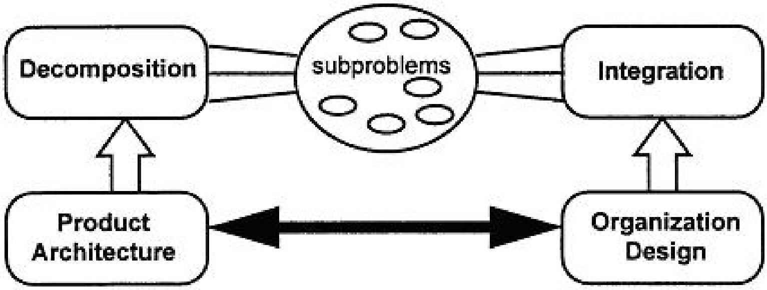


Fig. 4 Product architecture is tied to organization through the decomposition/ integration problem [31]

reaches a peak beyond which even small increase of entropy inexorably cause the breakdown of structure [46]. After structural breakdown commences, an increase in entropy nearly always leads to loss of complexity (fitness) [46]. However, beyond the critical point, loss is inevitable, regardless of the dimensionality and/or density of the system. Therefore every closed system can only evolve/grow to a specific maximum value of complexity. This is known as the system's critical maximum complexity. Close to criticality, systems become vulnerable, fragile and difficult to manage. The difference between the current and critical complexity is a measure of the overall "health" of the system. The closer to criticality a system is, the less healthy and therefore generally more risky it becomes.

2.4 Deriving a Team Based DSM from a Simulated Parameter Based DSM

Integrated product development (IPD) describes how tasks are interconnected and seeks to integrate the product process and organization (the network of the tasks) but does not provide adequate problem solving methodologies [49]. Choosing an integration scheme is critical in determining how efficient or how flexible a resulting problem solving architecture will turn out to be [49]. As a result, the ease of organizational partitioning and integration is tied to the nature of the product decomposition [31] (Figure 4). The product architecture and organization structure relationship can affect an enterprise in several dimensions, including architectural innovation [6].

The product architecture has a large influence on the appropriate structure of the product development organization since organizational elements are typically assigned to develop various product components [6]. Appropriately clustering interdependent design parameters can reveal a preferred integration of low-level activities into higher-level ones. Indeed, clustering may be a key to tying top-down, activity-based DSMs together with bottom-up, parameter-based DSMs [6]. To incorporate this knowledge into the product development system, we have developed a decomposition scheme founded on minimizing the complexity of the decomposed

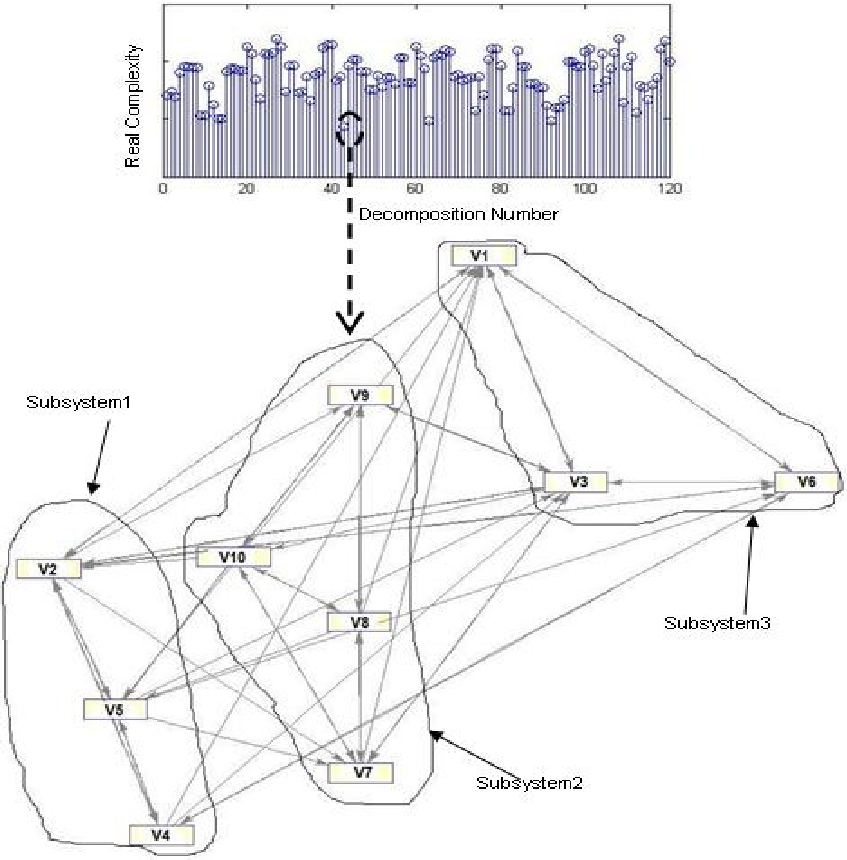


Fig. 5 The system is decomposed through minimizing real complexity

system (or real complexity, C_R) which is always more than or equal to the complexity of the system before the decomposition [24]. In Figure 5, 120 random decompositions of the system example in Table 1 are illustrated. In Table 2 the system after decomposition is shown.

The approach to organizational integration in IMMUNE is to directly derive the team based DSM from the simulated parameter based DSM. IMMUNE does not have any pre-specified organizational architecture – rather the developed organization integration scheme is determined by deriving the team based DSM from the simulated parameter based DSM. A direct mapping is used that forces the organization structure to mirror the product architecture. Therefore the above matrix is directly taken as the predicted team based DSM. The predicted team based DSM is used for two purposes: 1) to form the multidisciplinary and multi functional teams, and 2), to calculate the minimum and maximum process complexity.

Three teams responsible for the parametric problem solving of these variables must be chosen (same as the number of subsystems). These would normally be

Table 2 The variables of Table 1 are rearranged to form three subsystems

		Subsystem 1			Subsystem 2				Subsystem 3		
		V5	V4	V2	V10	V8	V7	V9	V6	V1	V3
Subsystem 1	V5	0	0.45	0.44	0	0	0.02	0.02	0.53	0.22	0.11
	V4	0.34	0	0.65	0	0	0	0	0.43	0.16	0.64
	V2	0.3	0.12	0	0	0	0.2	0.1	0.2	0.76	0.12
Subsystem 2	V10	0.01	0	0.18	0	0.8	0.1	0.9	0	0	0
	V8	0.01	0	0	0.1	0	0.2	0.4	0	0.01	0
	V7	0	0	0	0.3	0.45	0	0.1	0	0.12	0.02
	V9	0	0	0	0.5	0.2	0.7	0	0	0	0.15
Subsystem 3	V6	0	0.34	0.78	0	0	0	0	0	0.77	0.31
	V1	0	0	0.53	0	0	0	0	0.1	0	0.32
	V3	0	0	0.11	0.72	0	0.3	0.52	0.2	0.45	0

multidisciplinary and cross functional teams of people with different functional expertise and disciplinary knowledge responsible for working toward a common goal. It may include people from finance, marketing, operations, and human resources departments and different scientific disciplines. Typically, it includes employees from all levels of an organization.

Note that the team members or design agents at this stage are not yet explicitly defined but the likely interactions inside and across the teams are determined. From the above DSM a team based DSM is derived by summing up the amount of information exchange (dependency) of the design variables each is responsible for. In Table 3 the C_{Ti}'s denote the complexity of the interaction inside the teams, or they may be interpreted as the complexity that each team must internally deal with. The complexity based method is derived from the assumption that the complexity of an interconnected system may be represented by a single measure of that system which is termed a complexity measure. Figure 6 shows this simplified team based DSM. This predicted team based matrix reflects the amount of information exchange as well as the internal complexity of the problem that the design teams are dealing with.

From the above matrix a single measure of complexity of the entire problem that the network of design teams must deal with can be estimated by using the Ontix. This measure is equal to the real complexity of the decomposed system (C_R). Similar to the maximum and minimum complexity of the system before decomposition, real minimum and maximum complexity may be calculated for the above matrix ($\mathrm{C}_{Rmin,}\mathrm{C}_{Rmax}$).

Table 3 The predicted team based DSM for the entire system

-	Team1	Team2	Team3
Team1	C_{T1}	0.34	3.17
Team2	0.2	C_{T2}	0.3
Team3	1.76	1.52	C_{T3}

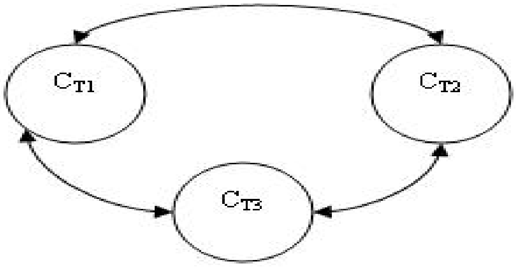
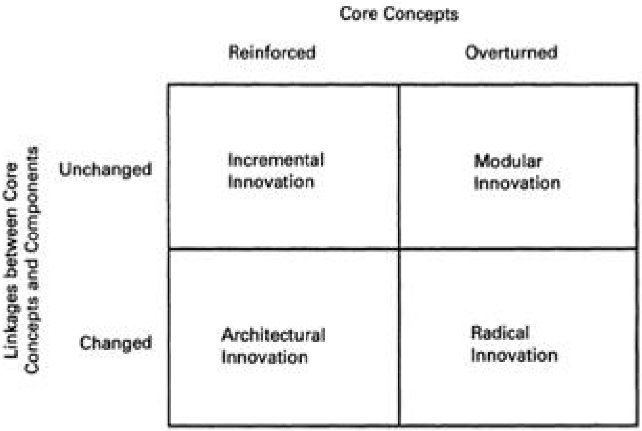


Fig. 6 The predicted team based map or the entire system

3 Radical Innovation

Henderson and Clark [32] demonstrated that there are different kinds of innovation as depicted in Figure 7 where innovation is classified along two dimensions. The horizontal dimension captures an innovation's impact on components (subsystems), while the vertical dimension captures its impact on the linkages between core concepts and components. Radical innovation establishes a new dominant design and, hence, a new set of core design concepts embodied in subsystems that are linked together in a new architecture. Incremental innovation refines and extends an established design. Improvement occurs in individual components, but the underlying core design concepts, and the links between them, remain the same. Modular innovation on the other hand, changes only the core design concepts without changing the product's architecture. Finally, architectural innovation changes only the relationships between modules but leaves the components, and the core design concepts that they embody, unchanged. We can say that radical innovation embodies both modular and architectural innovation.

An organization's communication channels, both formal and informal are critical to achieving radical and architectural innovation [32]. The communication channels that are created between these groups will reflect the organization's knowledge of the critical interactions between product modules. Organization's communication



Linkages between Core Concepts and Components	Core Concepts: Reinforced	Core Concepts: Overturned
Unchanged	Incremental Innovation	Modular Innovation
Changed	Architectural Innovation	Radical Innovation

Fig. 7 Different types of innovation [32]

channels will embody its architectural knowledge of the linkages between components that are critical to effective design [32]. They are the relationships around which the organization builds architectural knowledge.

Innovation processes in complex products and systems differ from those commonly found in mass produced goods [34]. The creation of complex products and systems often involves radical innovation, not only because they embody a wide variety of distinctive components and subsystems (modular innovation), skills and knowledge inputs but also because large numbers of different organizational units have to work together in a collaborative manner (architectural innovation). Here, the key capabilities are systems design, project management, systems engineering and integration [34]. Integration in complex system and product design is to make the solutions to subproblems compatible with each other and this is possible through innovation. The innovation that integrates the complex system must be radical innovation and creativity [62] and this is an emergent property of the entire system rather than the property of the sub-solutions to the individual subproblems [61]. A property that is only implicit, i.e. not represented explicitly, is said to be an emergent property if it can be made explicit and it is considered to play an important role in the introduction of new schemas [28]. The radical innovation and coherency in an engineered large scale system is emergent and obtained in a self-organizing fashion in a multi-agent environment. When designing self-organizing emergent multi-agent systems with emergent properties, a fundamental engineering issue is to achieve a macroscopic behavior that meets the requirements and emerges only from the behavior of locally interacting agents. Agent-oriented methodologies today are mainly focused on engineering the microscopic issues, i.e. the agents, their rules, how they interact, etc, without explicit support for engineering the required macroscopic behavior. As a consequence, the macroscopic behavior is achieved in an ad-hoc manner [69].

Creativity requires ad hoc communication in which the need to communicate often arises in an unplanned fashion, and is affected by the autonomy of the agents to develop their own communication patterns (see Leenders et al. [43]). It is thus obvious that, a fixed organizational structure with established patterns of communication is not capable of delivering new complex structures (products). Also Leenders et al. [43] showed that team creative performance will be negatively related to the presence of central team members (including brokers, mediators and facilitators) in the intra-team communication network. ‘Emerging’ properties and innovative organizational structures are required to coordinate between different design teams that lead to integration of the entire complex system with coherency.

3.1 Holistic Process Monitoring

When the inherent nature of a complex task is too large, a better solution is to create an environment in which continuous innovation can occur [1]. This can be accomplished through process monitoring: Bayrak and Tanik [4] reported that improving the design process that increases the product quality without increasing the design resources is possible by providing feedback to the designer to help him/her understand the nature of the design process. Therefore, the nature of the design becomes easier to analyze if there are metrics obtained from activity monitoring [4]. Since the design process of the complex systems by concurrent engineering is an *emergent process* [11], holistic metrics are required to monitor the design process. One such metric is the cognitive complexity of a process that is defined as the ability of a problem solver to flexibly adapt to a multidimensional problem space [42]. Cognitive complexity represents the degree to which a potentially multidimensional cognitive space is differentiated and integrated. A problem solver (a person, organization or a multi-agent system) with higher cognitive complexity is more capable of having creative (and holistically correct) outcomes [42].

We suggest measuring the cognitive complexity of a multi-agent design process as a function of the information exchange between the design agents. The main complication here is the way in which the information exchange is measured. Kan and Gero [35] suggested the use of entropy based measures for the evaluation of information content of a design agent’s interactions. We suggest using a fuzzy method by simply asking the design participants to tag qualitative and quantitative information content of their interactions with a single fuzzy variable, e.g. high, low, and medium, etc. These can then be defuzzified, which is the process of producing a quantifiable result in fuzzy logic, according to a simple fuzzy rule shown in figure 8.

An example of the produced fuzzy DSM and the procedure of estimating the cognitive process complexity for a design system comprised of three design teams is illustrated in figure 9. Four design agents are assumed to be in team 1, the internal interactions of which create the cognitive complexity of team 1 (C_{Ct1}). Having measured the cognitive complexity of all the design teams using Ontix (C_{Ct1}, $\mathrm{C}_{Ct2,}$ C_{Ct3}) of any instance, and knowing the amount of intra team information exchange, one can measure the cognitive complexity of the entire system (C_{CS}).

Note that the information mentioned here is *qualitative* and *quantitative* and therefore the proclivity of the design agents for more collaboration by the means

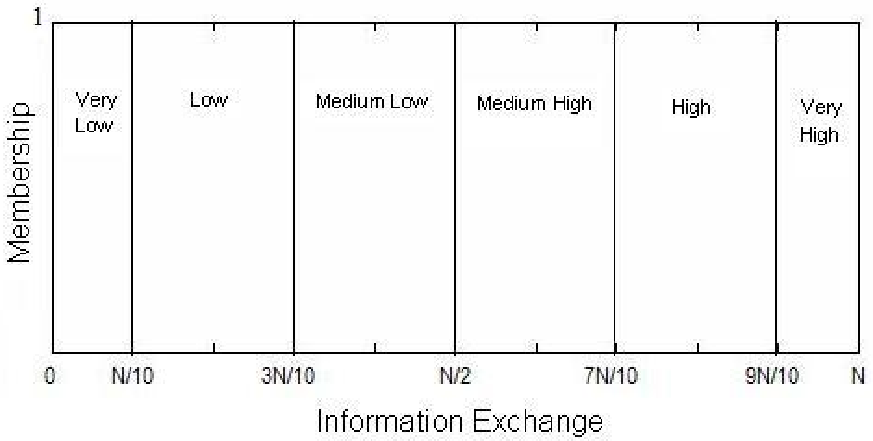


Fig. 8 A simple fuzzification scheme. *N* is the total number of parameters

of information exchange do not necessarily lead to more overall cognitive complexity of the design system. This is to say, unnecessary information exchange may lower the overall cognitive complexity. As such, by testing a sample of 44 new product development organizations, Leenders et al. [43] showed that the performance of innovation networks (innovation teams) have an inversely U-shape relationship to frequency of intra team cooperation. Considering this, one can conclude that the cognitive complexity of the entire design system must be upper and lower bounded for effective and efficient innovation networks.

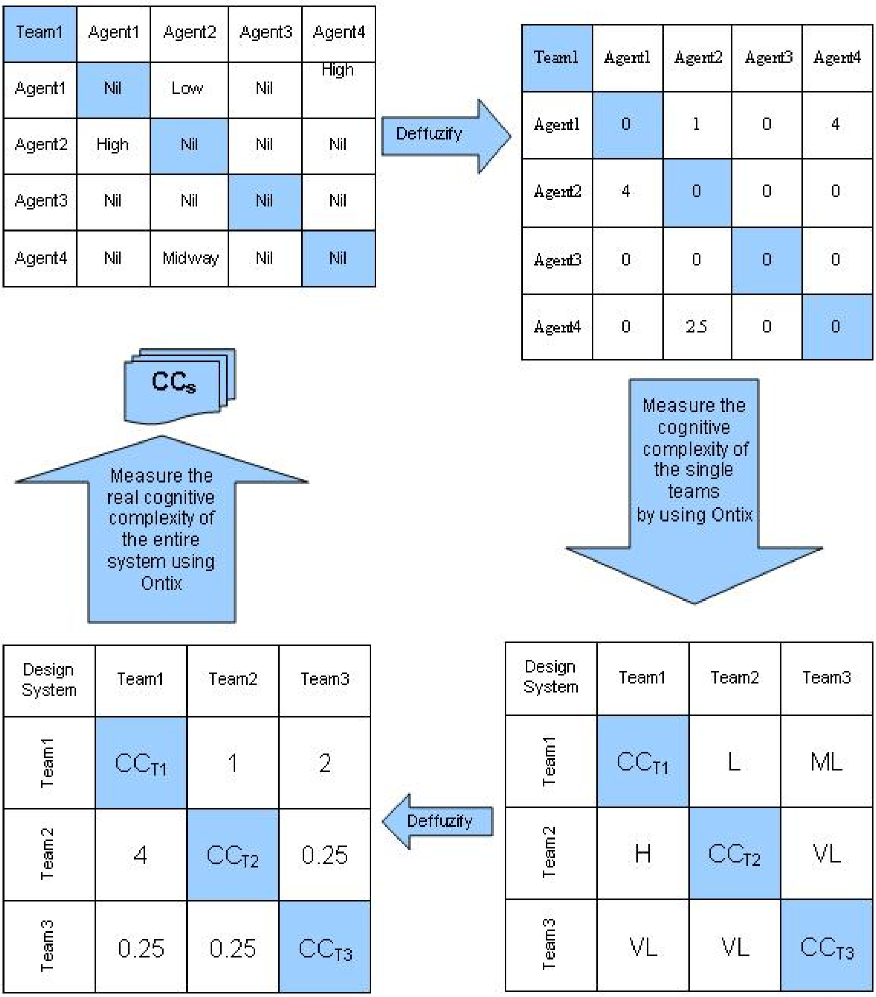
Bar-Yam [2] has stated that in order to solve a problem, the problem solver needs to have (cognitive) complexity greater than or equal to the problem complexity. This is to say, the cognitive complexity of the process must be equal or greater than the real (overall) minimum (irreducible) complexity of the problem.

$$C_C \geq C_{Rmin} \tag{1}$$

Also Chiva-Gomez [10] favoured a balanced participation of design players in the design decision making process, against increasing information flow between the design players to a maximum. It is obvious that the cognitive complexity of the process need not be more than the real maximum complexity of the problem. In fact, having more than the required information exchange can lead to creativity blocks [43] which can be termed a chaotic situation. Thus an upper bound for the cognitive complexity of the process is the real maximum complexity of the problem:

$$C_C \leq C_{Rmax} \tag{2}$$

Innovation in a multi-agent environment is the result of communication between social agents that happens in a self-organizing fashion and when the multi-agent system finds itself on the so-called edge of chaos [64]. When the cognitive complexity of the process is in between the real minimum and maximum complexity of the


Deffuzify
Measure the cognitive complexity of the single teams by using Ontix
Deffuzify
Measure the real cognitive complexity of the entire system using Ontix
CC_s

Team1	Agent1	Agent2	Agent3	Agent4
Agent1	Nil	Low	Nil	High
Agent2	High	Nil	Nil	Nil
Agent3	Nil	Nil	Nil	Nil
Agent4	Nil	Midway	Nil	Nil

Team1	Agent1	Agent2	Agent3	Agent4
Agent1	0	1	0	4
Agent2	4	0	0	0
Agent3	0	0	0	0
Agent4	0	2.5	0	0

Design System	Team1	Team2	Team3
Team1	CC_{T1}	L	ML
Team2	H	CC_{T2}	VL
Team3	VL	VL	CC_{T3}

Design System	Team1	Team2	Team3
Team1	CC_{T1}	1	2
Team2	4	CC_{T2}	0.25
Team3	0.25	0.25	CC_{T3}

Fig. 9 Measuring the cognitive complexity of the design process of any instance

problem, the design system might be on the edge of chaos but certainly not chaotic. Besides, for collaborative multi-agent systems with cognitive complexity less than the minimum complexity, the design process is certainly far away from the edge of chaos, thus the design systems does not have enough functionality to deliver radical innovation in an optimal and efficient manner. Extravagant and excessive practice of collaboration and cooperation has a negative effect on the design system by reducing the collective cognitive complexity ; as such, this condition strikes one as being chaotic. Chaos makes the design process fragile and susceptible and raises the design system effectiveness. It can be tested in figure 10 that design system's overall cognitive complexity increase only to a certain threshold by the proclivity of

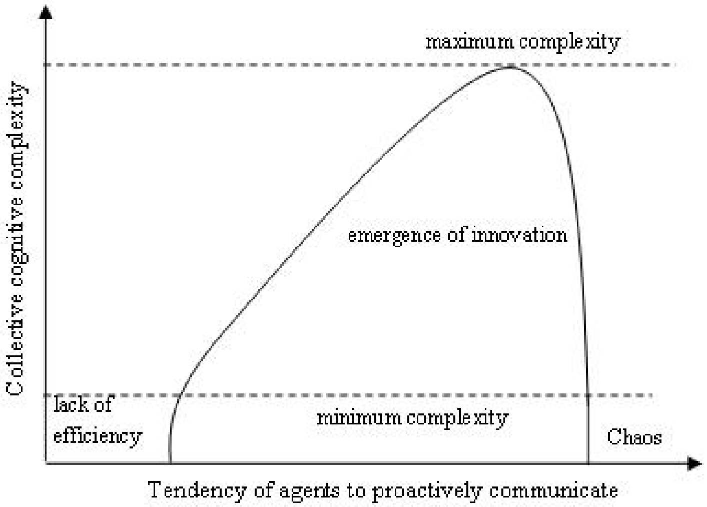


Fig. 10 Design process functionality versus process complexity

the design agents for exchanging design information. In order to ensure the health of the design process it is necessary to ensure that the overall cognitive complexity stays away from the maximum complexity and above the minimum. This way the real minimum and maximum complexity that are obtained using the initial Monte Carlo Simulation of the complex product (low level product knowledge) are used to monitor the efficiency and effectiveness (health) of the complex product design process. This may be achieved through monitoring the design process. The process monitoring here serves the purposes of meeting the design objectives (quality, cost, and lead time) by immunizing the design system against chaos and lack of effectiveness. This immunization enables the design system to integrate the complex system and product through utilization of radical innovation

3.2 Artificial Immune Systems (AIS)

According to Cohen [12] the immune system is a "computational strategy" to carry out the functions of protecting and maintaining the body. Cohen's maintenance role of the immune system requires it to provide three properties:

1. *Recognition*: to determine what is right and wrong.
2. *Cognition*: to interpret the input signals, evaluate them, and make decisions.
3. *Action*: to carry out the decisions.

These properties are provided via a cognitive strategy in which self-organization of the immune system is used to make decisions [66]. The stages correspond to the

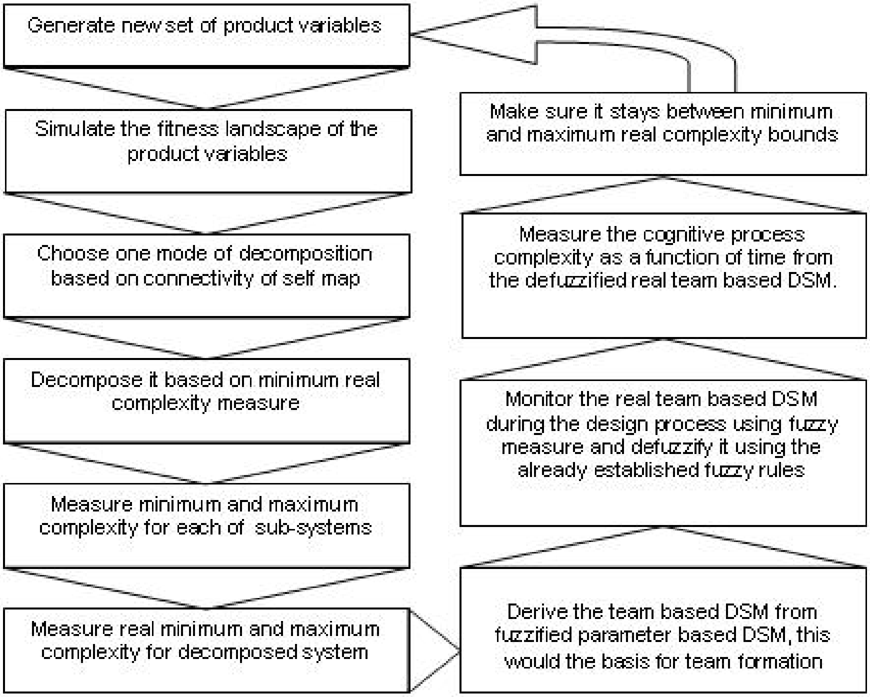

Generate new set of product variables
Simulate the fitness landscape of the product variables
Choose one mode of decomposition based on connectivity of self map
Decompose it based on minimum real complexity measure
Measure minimum and maximum complexity for each of sub-systems
Measure real minimum and maximum complexity for decomposed system
Derive the team based DSM from fuzzified parameter based DSM, this would the basis for team formation
Monitor the real team based DSM during the design process using fuzzy measure and defuzzify it using the already established fuzzy rules
Measure the cognitive process complexity as a function of time from the defuzzified real team based DSM.
Make sure it stays between minimum and maximum real complexity bounds

Fig. 11 Summary of the proposed immune algorithm for the design of complex systems

holistic control of the system, which is to immunize or ensure the realization of self-organization, by using a complexity measure:

1. *Recognition*: recognizing the lower and upper complexity bounds.
2. *Cognition*: to evaluate the instantaneous complexity of the system.
3. *Action*: to maintain this complexity in between the bounds at all times.

In the field of AIS an immune algorithm is a plan that determines how the components of the systems are going to interact to determine the system dynamics [66]. For example Dasgupta [16] examined various response and recognition mechanisms of immune systems and suggested their usefulness in the development of massively parallel adaptive decision support systems. Lau and Wong [41] presented a multi-agent system that could imitate the properties and mechanisms of the human immune system. The agents in this artificial immune system could manipulate their capabilities to determine the appropriate response to various problems. Through this response manipulation, a non-deterministic and fully distributed system with agents that were able to adapt and accommodate to dynamic environment by independent decision-making and inter-agent communication was achieved [41]. Ghanea-Hercock [29] maintained a multi-agent simulation model that could demonstrate self-organizing group formation capability and a collective immune response. He showed that the network of agents could survive in the face of continuous perturbations. Fyfe and Jain [27] presented a multi-agent environment in which the agents could manipulate their intensions by using concepts suggested by artificial immune systems to dynamically respond to challenges posed by the environment. Goel and Gangolly [30] presented a decision support mechanism for robust distributed systems security based on biological and immunological mechanism.

We define a system to be immune to chaos and capable of preserving its holistic self characteristics if its complexity is in between the minimum and maximum complexity bounds. The proposed immune algorithm provides collective immune responses for the engineering design of complex systems and is illustrated in Figure 11. This algorithm ensures the successful emergence of the complex product in a multi-agent design environment. This method is in accordance with the recent results that argue for flatter, organic organizational structures that enable workers to deal more effectively with dynamic and uncertain environments [33].

The next section describes the structure of a decision support system for complex system design that emulates this immune algorithm. It should be noted that the conceptual framework without any validation is presented here. The system can however be tested by using simulations in the context of game theory.

4 IMMUNE: A Collaborating Architecture

Real world concurrent engineering design projects require the cooperation of multidisciplinary design teams; individuals and multidisciplinary design teams work in parallel with various engineering tools that are located at different sites often for quite a long time [59]. At each instant, individual members may be working on different versions of a design or viewing the design from various perspectives (e.g. profitability, manufacturability, resource capability or capacity) at various levels of detail. To coordinate the design activities of various groups and guarantee good cooperation among the different engineering tools, a distributed intelligent environment is required [59]. Such an environment requires the utilization of *collaborating software* that involves the integration and coordination of relatively independent, self-contained software systems that are able to work together effectively on their own [14]; "*collaborating software*" is very different from "collaboration software," where the software is used to facilitate the interaction among human participants rather than to provide an automated environment where software—and potentially human—entities work together in order to perform complex activities [14]. For the effective development of collaborating software Corkill [14] identified six main challenges:

1. *Representation:* To enable system components and modules to understand one another.
2. *Awareness:* to render modules aware when something relevant to them occurs.

3. *Investigation:* helping modules to quickly find information related to their current activities.
4. *Interaction:* to create modules that are able to use the concurrent work of others while working on a shared task.
5. *Integration:* to combine results produced by other modules.
6. *Coordination:* ensuring that modules focus their activities on the right things at the right time.

All of these challenges, except for the representation, are addressed in IMMUNE and presented in this chapter. So far two main approaches have been considered for the design of a collaborating environment: Blackboard architectures and multi-agent architectures. Corkill [13] presented the following metaphor to describe Blackboard-based problem solving: "imagine a group of human specialists seated next to a large blackboard. The specialists are working cooperatively to solve a problem, using the blackboard as the workplace for developing the solution. Problem solving begins when the problem and initial data are written onto the blackboard. The specialists (knowledge sources) watch the blackboard, looking for an opportunity to apply their expertise to the developing solution. When a specialist finds sufficient information to make a contribution, she records the contribution on the blackboard, hopefully enabling other specialists to apply their expertise. This process of adding contributions to the blackboard continues until the problem has been solved". Each problem solving expert is designed to independently contribute specialized knowledge required to solve one aspect of the overall problem. The sequence of the contributions of these experts is not determined *a priori* but is instead based on the current state of the solution and the selection of the most applicable and effective expert for solving the associated problem part [53]. As such, the blackboard model of problem solving is a highly structured case of opportunistic problem solving [53].

Multi-agent systems, on the other hand, have the following problem solving characteristics [14]:

- Distribution (no central data repository)
- Autonomy (local control)
- Interaction (communication and representation)
- Coordination (achieving coherence in local control decisions)
- Organization (emergent organizational behavior)

Multi-agent System architectures are expressed as the pattern of relationships amongst agents [59]. Two kinds of relationships may be assumed between agents: Control relationships and Collaboration relationships [59]. A Control relationship relates to the degree of autonomy which an agent possesses. An agent whose goals, plans and/or actions are prescribed by the imperatives of another agent(s) has little autonomy. In a Collaborating relationship however, the agents involved are free to accept, reject or modify goals, plans or actions proposed to them. In theory, a truly open multi-agent system need not have any predefined global control. An example of such architecture is that of DIDE [57].

Shen et al. [59] have identified three different architectures for agent based systems: hierarchical architectures, federated architectures and autonomous agent architectures. Each architecture has particular strengths for specific applications, and choosing the right one involves matching requirements to capabilities. Hierarchical architectures consist of semi autonomous agents with a global control agent dictating goals/plans or actions to the other agents. Systems with a global blackboard data base are hierarchical architectures.

Because hierarchical architectures suffer from serious problems associated with their centralized character, federated multi-agent architectures are increasingly being considered as a compromise solution for industrial agent based applications, especially for large scale engineering applications. A fully federated agent based system has no explicit shared facility for storing active data; rather, the system stores all data in local databases and handles updates and changes through message passing. They mostly use local control regimes referred to as facilitators, brokers and mediators.

In general, collaborating environments use two different techniques to manage complexity: *abstraction and decomposition* [3]. Abstraction simplifies the description or specification of the system, by representing the same problem from different viewpoints and at various levels of detail. Decomposition, however, breaks a single problem into many smaller problems. According to Bar-Yam [3] each of these two techniques, when dealing with complexity suffers from inefficiency. Abstraction assumes that the details to be provided to one part of the system (module) can be designed independent of the details in other parts [3]. Decomposition incorrectly assumes that a complex system behavior can be reduced to the sum of its parts [3].

The Blackboard architecture, however, utilizes abstraction and solves problems through iteration. Because it uses a global memory, blackboard architectures are able to maintain the focus of attention of different knowledge sources asynchronously on different abstraction levels within this memory; the main point is that the knowledge sources do not communicate with each other directly and communication is solely done through the blackboard. This allows for asynchronous communication and thus blackboard systems suit loosely coupled problems [14]. On the other hand, multi-agent systems, due to their ability to interact autonomously, can reach to high overall cognitive complexity to solve densely interconnected problems without the need for a global memory (integrator). Collaboration in multi-agent environments can be asynchronous and is not restricted to one abstraction level; therefore agents must be allowed to focus on different aspect of the problem.

According to Corkill [14] a quarter-century of blackboard-system experience and more than a decade of multi-agent system development have produced a strong baseline of collaborating-software technologies. The next generation of complex, collaborating software applications must span the entire design space of Figure 12 to enable development of high performance, generic collaborating-software capabilities. This is the motivation behind the design of the presented architecture, IMMUNE, which combines agent based and blackboard technologies. This rare approach was first introduced by Lander et al. [39], in which they proposed the use of agent based blackboards to manage agent interactions [59]. Their model

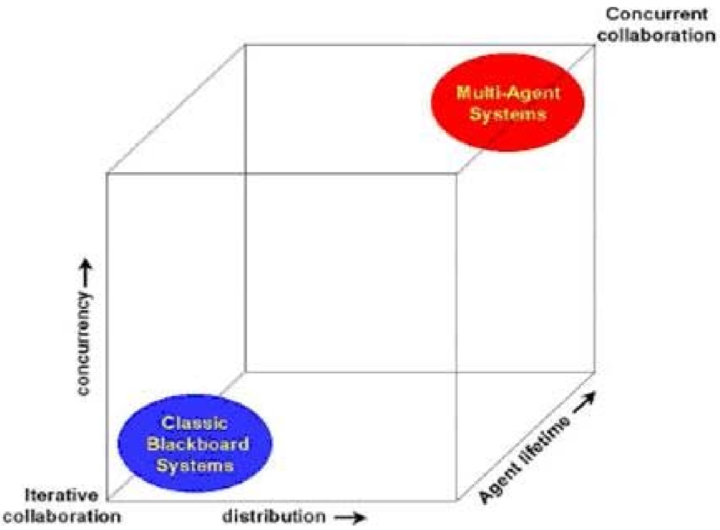


Fig. 12 Collaborating environment comparison [14]

contained multiple blackboards, used as data repositories for each group of agents. Along with design data, tactical control knowledge could be represented in the shared repository, enabling reasoning about the design itself [59]. SINE [5] was another agent based blackboard platform that used a single global blackboard to record the current state of the design. Even though agents could exchange messages directly, design data could flow through the blackboard, and it was accessible to all agents [59].

Our proposed architecture (IMMUNE) is an agent based blackboard system that uses a flat federated architecture. All the agents are grouped into virtual teams or coalitions. There is no local controller for coordination in between the coalitions. IMMUNE uses a global blackboard to save the current state of the design and to facilitate asynchronous communication between agents through the blackboard by saving the complete solution and partial solutions in different abstraction hierarchies.

The primary purpose of designing this architecture was to incorporate the complexity science into the collaborating software paradigm. Lissack [44] demonstrated that since both organization science and complexity science deal with uncertainty, it is important to combine the two. This marriage of the two sciences allows for having an autopoietic view of organization. Autopoietic systems theory analyse systems as having self-productive, self-organized, and self-maintained nature [20]. The main characteristic of IMMUNE is to emulate complexity measures of the product and process which enables the manifestation of autopoietic characteristics.

The control source of the proposed blackboard does not dictate the pattern of cooperation between agents, thus allowing autonomy in the interaction. It however does monitor the complexity of the system at two levels: inside the coalitions and in between the coalitions at the same abstraction level (we refer to this as a layer).

The agents are designed to react to the information that they receive from the control source about the complexity of the coalitions and layers. Adding or eliminating agent(s) from the design system is possible in IMMUNE, making it an open architecture.

4.1 Blackboard Architecture

The Blackboard Database is a hierarchical and partitioned global memory space that acts as a central storage area for holding problem solving data, information and partial solutions that represents the problem to be solved [15]. The blackboard provides a common data structure that acts as an interface to agents (or knowledge sources in standard blackboard systems) allowing them to read the problem data and alter the state of this data when necessary, thereby effecting an incrementally improved solution to the problem [53]. The higher abstraction levels relate to the more intrinsic characteristics of the product, which has to do with the main functions and performances of the product. The lower abstraction levels locate the variables that describe the detailed functionalities of the product. Bearing this in mind, it is clear that the first abstraction level in the hierarchy is like a seed that all the solutions of all other abstraction level variables depend heavily upon. The seed must reflect the most important and general functions of the product. Each abstraction hierarchy is decomposed according to one of the modes that will be described in Section 5. Abstraction levels are introduced on the blackboard gradually according to the GDDI model of design. The IMMUNE blackboard contains the design variables, their interactions (derived from simulation) and the agents' proposed solutions.

4.2 Control Source

Typically blackboard architectures provide a control mechanism called a control source to coordinate the use of knowledge sources in a consistent and effective manner [53]. The control source determines which knowledge sources should make a contribution to the solution, when they should do so, and what part of the solution should be the focus. In IMMUNE, however, the agents decide their focus of attention in a manner described in the next section. The control source of IMMUNE comprises several agents with distinct tasks, all of which can be computationally modeled:

- *Decomposition agent*: decomposes the generated problem on the main blackboard according to the connectivity of the problem. Important control features that affect the entire system's performance can be incorporated in this agent's knowledge namely the number of subsystems, and decomposition mode. The former is beyond the scope of this paper although the later is discussed comprehensively in Section 5.

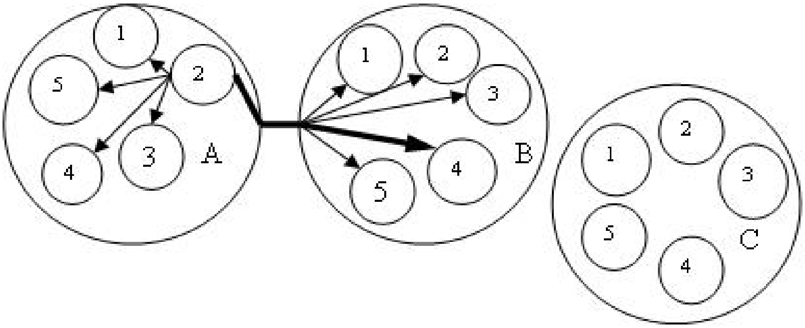


Fig. 13 Shared mail boxes for coalitions

- *Composition Agent*: groups the agents based on their bids for the problems in coalitions using the contract net protocol. A composition agent contains the map of all the agents, their characteristics and types of expertise.
- *IT manager*: Sets up the LAN and communications channels of the dispersed agents for each abstraction level.

  All the agents in the same coalition must be visible to each other, meaning that the messages that one agent receives are made visible to all team members. This may be thought of as a shared mailbox for each coalition. For example, Figure 13 shows that when a message is sent from agent 2 in coalition A to agent 4 in coalition B, it would be visible to all the members of these two coalitions.
- *Simulation Agent*: Performs Monte Carlo Simulation to generate the design space fitness landscape. It comprises a Monte Carlo Simulation software package and a human operator. It gathers information about the conditional probability distribution of the design variables from the agents that generate them. As the criteria for the termination of the generation stage of a given abstraction level can be based on the self complexity of the abstraction level, the composition agent must be able to dynamically simulate the fitness landscape as the new entries (design variables) appear on the blackboard for a given abstraction level.
- *CEO (complexity evaluator and observer)*: This agent announces the termination of the generation stage for a given abstraction level as soon as the complexity of the level reaches a certain threshold. This threshold is a control feature of the entire system. This agent also monitors the design process. It has an embedded blackboard on which all the communications between the agents and in between the agents and blackboard are recorded (Figure 14). The design agents can only write on this blackboard but there is no necessity for them to be able to read it. The communication arrows on this blackboard must have a tag that represents both the qualitative and quantitative weight of transferred information. Based on these maps of the system which vary regularly over time, the CEO measures the instantaneous cognitive process complexity of each coalition (a team of agents) and layer (in between the coalitions of one abstraction level). The CEO measures the lower and upper complexity bounds for the coalitions and also upper and lower complexity bound for layers. The control source must contain knowledge

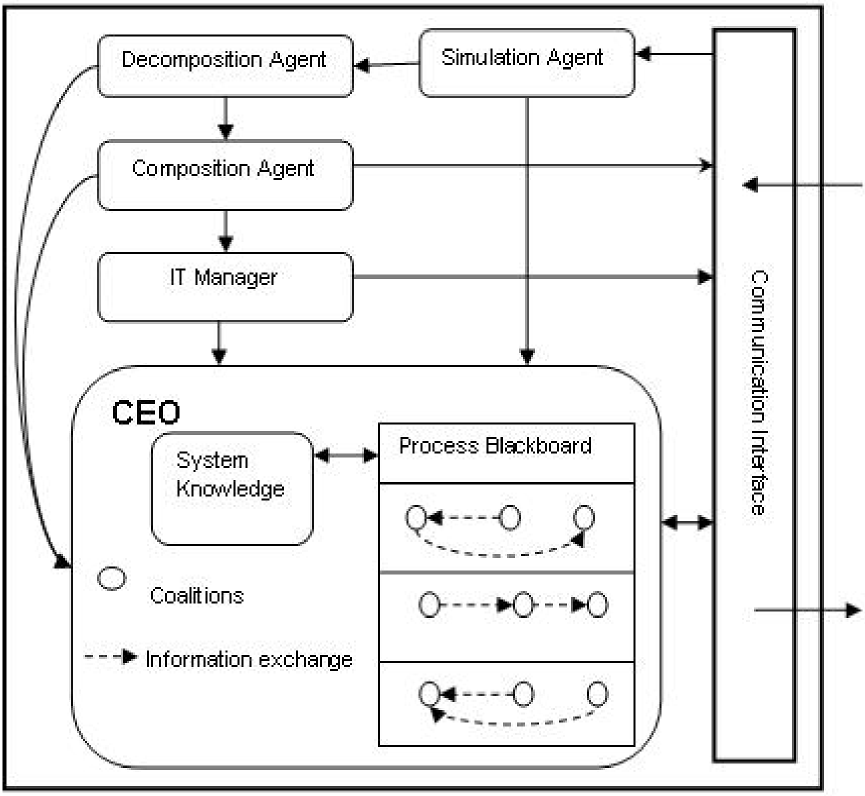


Fig. 14 The control source structure of IMMUNE

of the different types of resource agents in terms of their capabilities, functionalities and discipline knowledge. If an agent is being recruited to the system it must register all its characteristics with the composition agent of the control source.

The control source may be fully computational and may not need any human intervention to proceed with its tasks. The design process begins with the control source broadcasting notices to all agents with regard to the generation of new design variables. The agents place their entries on the blackboard in the specified abstraction level. This is the generation stage of the GDDI cycle. As mentioned before, the termination of the generation stage in each abstraction level, can be conditional to the self complexity of the level. The first set of initial design variables act like a seed on the blackboard that would gradually evolve to other design parameters at other abstraction levels. The control source, then, whether by itself or through knowledge sources, is in charge of simulating the fitness landscape (or design space) corresponding to these sets of design parameters and the extraction of the self map for the design parameters. After this, the control source decides on the number of sub problems, and the decomposition mode, decomposes the self by considering the number of active design agents (design resources) and the real complexity. The control source then clusters the design agents into virtual teams and distributes the subproblems to them. The design agents within the virtual teams solve the problems cooperatively. They send the results back to the blackboard, and negotiate the conflicts with the other groups until they reach a resolution.

Using the common practices of sequential engineering would lead to starting a new cycle (GDDI cycle) after all activities of the previous cycle are performed and the results are finalized. However, the concurrent engineering principle of overlapping the activities to shorten the design lead time may be applied here. Therefore agents must be allowed to introduce their proposed design variables on the blackboard (problem generation), however the decomposition and distribution stages start only when the CEO supposes an abstraction level as having reached a certain complexity threshold. Since agents can be cloned to perform different tasks, it is possible that two or more abstraction levels could simultaneously be at different stages within the GDDI cycle. In order to reduce the complexity of the entire design system, we propose that the design agents of different abstraction levels be allowed to communicate only through the blackboard.

4.3 Agents Structures

In artificial intelligence, an intelligent agent observes and acts upon an environment, as a rational agent: an entity that is capable of perception, action and goal directed behaviour. Such an agent might be a human, computer code or an embedded real time software system. The internal architecture of an agent is essentially the description of its modules and how they work together [59]: agent architectures in various agent based systems (including agent based concurrent design and manufacturing systems) range from the very simple (a single function control unit with a single input and output) to very complex human like models. The agents in IMMUNE are Single Function Agents (SiFAs) which were developed in the AIDG research group [22] to investigate concurrent engineering design problem solving using multi-agent architectures. It involves multiple agents that cooperatively produce a solution when the task of the entire system is decomposed into many, very small subtasks; where exactly each one of these is assigned to an individual agent [22]. Every agent now has one function to perform, that is, to execute its subtask. Agents have their own point of view that represents the expertise of the agent; which might be cost, strength, manufacturability etc [22]. In IMMUNE, SiFAs have three functions: 1) to generate the design variable; 2) to estimate the values of the design variables; and 3) to evaluate the solutions of other agents from their own point of view, verifying the existence of any conflicts. SiFAs are collaborative agents (also called interacting or social agents) that work together to solve problems. The joint expertise of collaborative agents is applied to ensure that the overall design is consistent [59].

Coherence is a global (and regional) property of the MAS that could be measured by the efficiency, quality, and consistency of a global solution (system behavior) as well as the ability of the system to degrade gracefully in the presence of local failures [65]. Coherency is about the ability of the MAS's to "cope" with problem integration. Several methods for increasing coherence have been studied, all of which relate to the individual agent's ability to reason about the following questions: who should I interact with, when should I do it, and why? Sophisticated individual agent reasoning can increase MAS coherence because each individual agent can reason about non-local effects of local actions, form expectations of the behaviour of others, or explain and possibly repair conflicts and harmful interactions [65]. On this basis, four different agent architectures have been discussed in the literature: reactive agents (also known as behaviour based or situated agent architectures), deliberative agents (also called cognitive agents, intentional agents, or goal directed agents), collaborative agents (also called social agents or interacting agents), and hybrid agents [59].

Reactive agents are passive in their interactions with other agents. They do not have an internal model of the world and respond solely to external stimuli. They respond to the present state of the environment in which they are situated. They do not take history into account or plan for the future [65]. Through simple interactions with other agents, complex global behavior can emerge. In reactive systems, the relationship between individual behaviors, environment, and overall behaviour is not understandable [59]. However, the advantage of reactive agent architecture is simplicity [59].

Deliberative agents use internal symbolic knowledge of the real world and environment to infer actions in the real world. They proactively interact with other agents based on their sets of Beliefs, Desires and Intentions (BDI system). These agents perform sophisticated reasoning to understand the global effects of their local actions [65]. Consequently, they have difficulties when applied to large complex systems due to the potentially large symbolic knowledge representations required [65]. Shen *et al.* [59] identified collaborative agents as a distinct class of agents that work together to solve problems; communication between them leads to synergetic cooperation, and emergent solutions.

Hybrid architectures are neither purely deliberative nor purely reactive [65], and the agents in IMMUNE have a hybrid architecture (Figure 15). According to Sycara [65] hybrid agents usually have three layers: at the lowest level in the hierarchy, there is typically a reactive layer, which makes decisions about what to do on the basis of raw sensor input. This layer contains the self knowledge that is the knowledge of the agent about itself including physical state, location, and skills, etc. [59]. The agent's self knowledge is used to participate in tasks and reply to other agents as well as control source requests about its competence.

The middle layer, typically abstracts away from raw sensor input and deals with a knowledge-level view of the agent's environment, often making use of symbolic representations [65]. This layer contains two types of knowledge: domain knowledge and common sense knowledge. The domain knowledge is the description of the working projects (problems to be solved), partial states of the current problem, hypothesis developed and the intermediate results [59]. The common sense knowledge enables the agent to emulate and make sense of the cognitive complexity measure

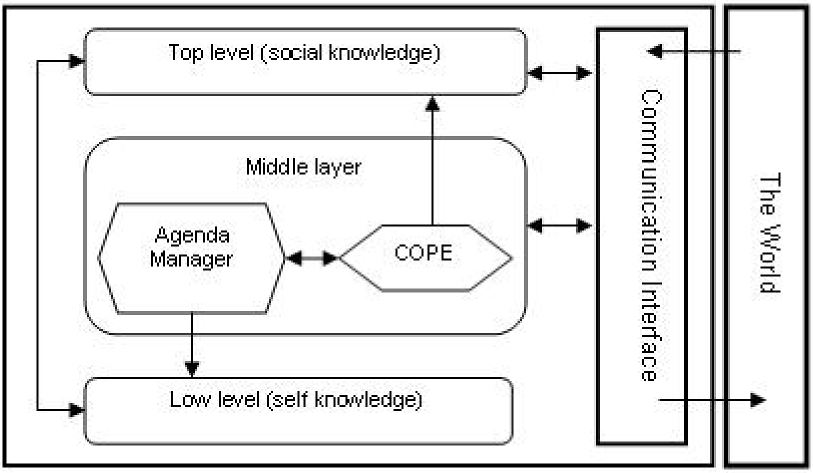


Fig. 15 Agent structure of IMMUNE

of the environment that is reported by the CEO. The middle layer has two modules that are in contact with backboard and the CEO and correspond to two major responsibilities:

1. *Deciding on the focus of attention and reporting it to the lower layer:* in this, the middle layer acts as an *agenda manager* that has been used in many blackboard systems such as HEARSAYII [8] with the difference that in these systems a central agenda manager has been in charge of maintaining the focus of attention for the entire set of knowledge sources (agents). Generally agenda managers are data driven (what is present on the blackboard) and its operation leads to opportunistic problem solving [8]. The agenda manager chooses the focus of attention of the agent on different problems at different abstraction levels. It may shift the focus of attention of the agent from one abstraction level to another depending on the status of the problems and partial solution on the blackboard. The main reason for using agendas for control is to speed up the process of problem solving, and for reducing agent idle time [8]. The agenda manager in the middle layer emulates the domain knowledge and regularly monitors the blackboard to maintain its domain knowledge.
2. *Maintaining the cognitive complexity of the coalitions in the focused level in the appropriate range:* This module contains the common sense knowledge (or symbolic knowledge of the world) which (in IMMUNE) is the collective cognitive complexity of the environment being broadcasted by the CEO. We refer to this module as COPE (Complexity Oriented Problem Evaluator) that can make sense of the environment's cognitive complexity by comparing it to the maximum and minimum complexity that is determined by CEO. COPE is a goal driven module and communicates with the agent's upper layer. To increase the complexity of the environment COPE informs the upper layer of the agent to socialize and

> collaborate more actively. To decrease the complexity of the environment the upper layer of the agent becomes more passive by only reacting to the incoming information from the control source and other agents and avoiding proactive engagement in the design process" In this way COPE may provide immunity from agent overreacting or under acting in the environment. Also COPE can dictate to the upper layer to choose different conflict resolution schemes that are more passive like constraint relaxation to reduce the complexity. Conversely, active negotiation techniques can be used to increase the cognitive complexity of the environment when there is conflict with another agent's solutions.

The uppermost level of the architecture handles the social aspects of the environment [65]. This layer contains the social knowledge and is in charge of coordination with other agents. It reports its information exchanges to the process blackboard of the CEO.

5 Implementation and Overall Behavior

The control source of IMMUNE is active throughout the entire design process. The design process starts with the generation of an initial set of product variables upon a notification from the control shell to single agents to introduce their entries on the blackboard. This set of product variables act as a seed representing the highest abstraction hierarchy of the problem space. The seed might be a rough guess of what needs to be done (Figure 16). The simulation agent of the control source simulates the fitness landscape of the generated problem space and is in charge of gathering all the required information (for simulation) from the design agents. The decomposition agent of the control source decomposes the set of generated variables and calls for the design agents' bids to participate in solving them. The agents announce their interest back to the control shell by weighting their interest in solving each individual design variable or estimating the value of a design constraint. The composition agent assigns each individual parameter to a single design agent. So far the process performs the same as SINE, which was a support platform for single function agents [5]. However, in IMMUNE the single function agents are also grouped into virtual teams (coalitions). The composition agent announces the coalitions' formats; this issue is discussed in detail in Section 5.2. The IT manager is in charge of setting the shared mail boxes for each coalition. The problems are solved by the design agents and the results are sent back to the blackboard. The CEO agent of the control source estimates the minimum and maximum process cognitive complexities that are exactly the minimum and maximum complexity of the problem. It monitors the design process cognitive complexity arising from the collaboration of the design agents during this last stage. If all the solutions are prepared, the virtual groups are dismantled and collaboration process is stopped. The next set of design variables is introduced and the cycle continues.

One possible drawback of this approach is that the design agents might be idle for a while until a task is assigned to them. To rectify this problem we introduced

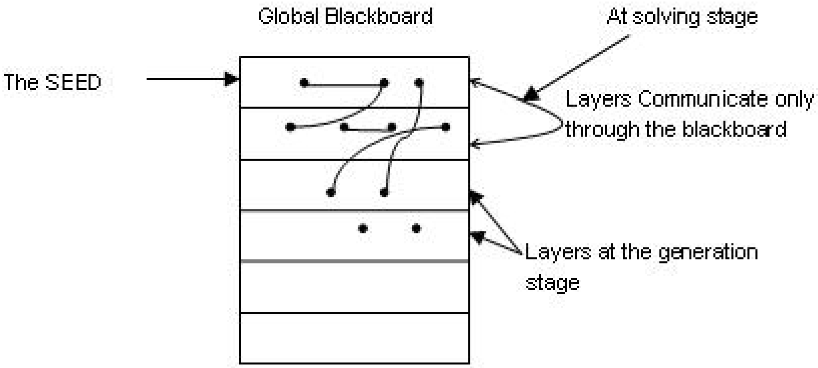

Global Blackboard
At solving stage
The SEED
Layers Communicate only through the blackboard
Layers at the generation stage

Fig. 16 The design process at different abstraction levels may run simultaneously

an agenda manager for each individual design agent. The agents are allowed to introduce new product variables at any abstraction level at any time during the design process. They may also be cloned to solve two different design variables related to different design groups or in the same design group. Also it is possible that the entire process of two abstraction levels be at run at the same time. To reduce the complexity of the system we propose that agents in the same abstraction level be allowed to communicate directly but agents that are working in different abstraction levels be allowed to communicate only through the blackboard. In order to facilitate this process, the simulation agent must respond dynamically. This notion is further discussed in the next section.

5.1 Structuration: Adaptive Organization Structure with Virtual Cross-Functional Teams

An important research question in the field of organization design is how to constitute cross-functional teams (See Browning, [6]). Coordination schemes are needed to direct the design process so that a design solution is sought in a way that accommodates the team interactions [9]. The productivity of design teams depends to a large extent on the ability of its members to tap into an appropriate network of information and knowledge flows [37]. Utilizing cross-functional teams that adapt the organization structure to the task structure is one way to address these situations [6]. Browning [6] has argued that appropriately clustering design parameters can reveal a preferred integration of low-level activities into higher-level ones and that indeed clustering is a key to tying top-down, activity-based DSMs together with bottom-up, parameter-based DSMs. At each round of the GDDI cycle the organization (team configuration) must be adapted to the generated tasks and reconfigured. Giddens (see [19]) has studied this problem extensively in a sociological context and developed a theory of Structuration. This theory is formulated as “the production and reproduction of the social systems through members’ use of rules and resources in interaction”. DeSanctis and Poole [19] adopted Giddens’ theory to study the interaction of groups and organizations with information technology, and called it *Adaptive Structuration Theory*. Structuration Theory, thus deals with the evolution and development of groups and organizations. The theory views groups or organizations as systems with (“observable patterns of relationships and communicative interaction among people creating structures”). Systems are produced by actions of people creating structures (sets of rules and resources). It is useful to consider groups and organizations from a structuration perspective because doing so [19]:

1. Helps one understand the relative balance in the deterministic influences and willful choices that reveal groups’ unique identities.
2. Makes clearer than other perspectives the evolutionary character of groups and organizations.
3. Suggests possibilities for how members may be able to exercise more influence than they otherwise think themselves capable of.

Virtual teams (or coalitions) are groups of individuals collaborating in the execution of a specific project while geographically and often temporally distributed, possibly anywhere within (and beyond) their parent organization [43]. Virtual teams work across boundaries of time and space by utilizing modern computer-driven technologies; as an instrument of team design, Information Technology (IT) is used for creating interdependent relationships by actively shaping and reshaping interdependencies and the communication structure of the virtual teams [43]. As these are altered, consequently, so is the team’s productivity [43] and creativity [18] which are related to the creation of the appropriate information flow between the design teams. Adaptive Structuration can be implemented with less effort by using virtual teams. This is so, because the creation of interdependencies within and between the virtual teams is arguably easier than in a conventional team [18], and so the management of creativity is easier in virtual organizations [43].

The idea of embedding different knowledge sharing patterns amongst the design agents and design teams of one system has been to date, considered by several systems researchers ([9, 55, 72, 47, 57]). MetaMorph [47] was a federated architecture and could dynamically adapt to emerging tasks and changing environment. In MetaMorph, core collaboration mechanisms are based on task decomposition and dynamically formed agent groups [59]. The agents with various knowledge and utility are clustered into virtual groups or coalitions. In MetaMorph, resource agents are cloned as needed for concurrent information processing [47]. These clone agents are included in the virtual clusters.

Shen and Norrie [58] argued that knowledge in modern manufacturing must be well organized and should be able to be flexibly applied to different kinds of applications. They used three different types of knowledge sharing architectures primarily introduced by Tomiyama et al [67]. A situation with independent knowledge bases

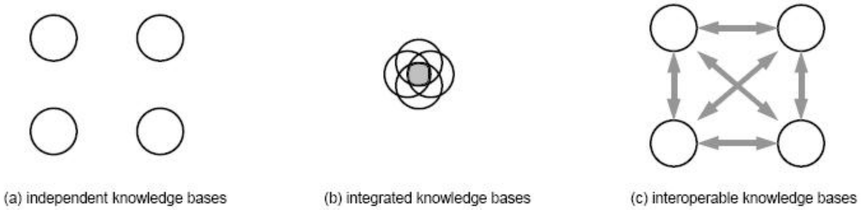

(a) independent knowledge bases
(b) integrated knowledge bases
(c) interoperable knowledge bases

Fig. 17 Knowledge sharing architectures [67]

is depicted in Figure 17a. In this case, the 'strength' of knowledge is just a sum of each of the independent knowledge bases [58]. Integrated knowledge bases can be represented as in Figure 17b. Here, the knowledge bases can be applied to various situations and the 'strength' of knowledge is near maximum [58]. In Figure 17c independent knowledge bases can communicate and form an interoperable situation, although the 'strength' of knowledge might be weaker than that in Figure 17b [58]. The entire knowledge base is a federated set of loosely coupled intelligent agents.

Zhang [72] classified types of problem solving among human experts in four predominant categories according to their interdependencies:

(a) *Horizontal cooperation* is where each expert in the cooperative group can get solutions to problems without depending on other experts, but if the experts cooperate, possibly using different expertise and data, they can increase confidence in their solutions. For example, cooperation between doctors when diagnosing problem patients is often a form of horizontal cooperation. Consultation and comparison of opinions add significantly to the value of the confidence of the final diagnosis.
(b) *Tree cooperation* is where a senior expert depends on lower-level experts in order to get solutions to problems. For example, a chief engineer's decisions often depend on the work of junior engineers.
(c) *Recursive cooperation* is where different experts mutually depend on each other in order to get solutions to problems. For example, in order to interpret geological data, geological experts, geophysical experts, and geochemical experts often depend on each other in a recursive way. That is, there is a mutual dependence in that a geophysicist may ask a geologist to perform a subtask which in turn requires performance by the geophysicist of the same sub-subtask. (NB: tree cooperation is a special case of recursive cooperation.)
(d) *Hybrid cooperation* is where different experts use horizontal cooperation at some level in an overall tree or recursive cooperation. On the other hand, they could equally use tree or recursive cooperation at some point in an overall horizontal cooperation.

Rosenman and Wang [55] introduced the Component Agent-based Design-Oriented Model (CADOM) for collaborative design. This was a dynamic integrated model, using an integrated schema to contain data for multiple perspectives, but

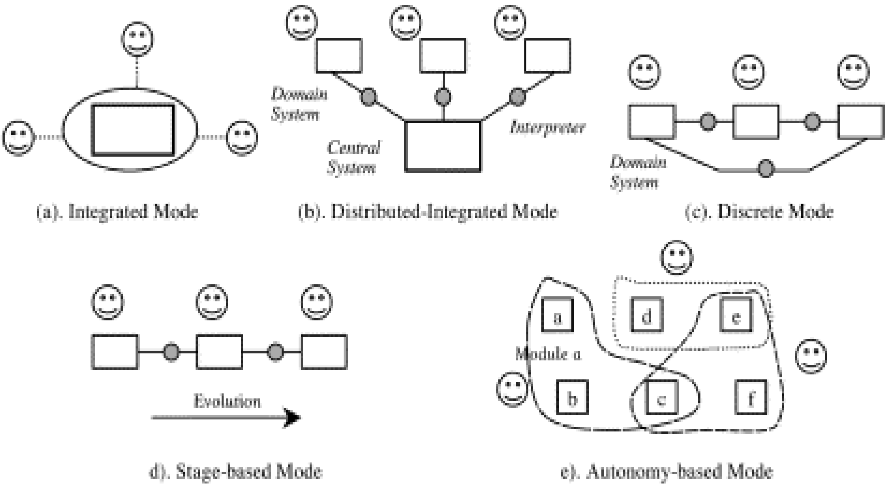


Fig. 18 System modes for collaborative design systems [55]

also with flexibility to support dynamic evolution. They recognized five types of modeling mechanisms for a collaborative design environment (Figure 17).

(a) Integrated mode (Figure 18a): This is an integrated CAD system which works as a sharable server for all users using an integrated data model and a central management mechanism. The distributed users register with the main host and operate the system remotely. However, such an integrated system does not seem able to meet the complex design requirements needed in a multi-disciplinary environment. For example, each item on the system will be communicated to all users.

(b) Distributed-integrated mode (Figure 18b): in this mode, distributed designers usually have their own domain systems along with a central service module called a sharable workspace.

(c) Discrete mode (Figure 18c): this is a fully distributed system, where there is normally no central module but simply a set of distributed domain systems with discrete models and management mechanisms. The most obvious feature of this mode is its flexibility, without a central control unit, but many model interpreters are required between different domain systems.

(d) Stage-based mode (Figure 18d): In this mode a base model is set up at the first stage, and all subsequent models are derived from the base model. Some internal mechanisms are provided to control this evolution process. This evolutionary system solves the flexibility of a system, but it requires a great deal of AI work to develop the system.

(e) Autonomy-based mode (Figure 18e): This is based on the concept of autonomy, in which each model is implemented as a distributed set of knowledge sources representing autonomous, interacting components.

5.2 Global Decomposition Strategies and Modality

According to Chen and Li [9], in order to enable dynamic structuration and adaptability, two preconditions must be satisfied:

1. The relevant design attributes, functions and variables necessary to formalize a design problem have been identified.
2. The interacting relation existing in teams is prescribed a priori in seeking a multi-team design solution.

In Section 2 we suggested that prior knowledge of the interactions (driven from the simulated parameter based DSM) should be used in measuring the process complexity, instead of being used for forced communications through channels and filters. The reason to use this democratic method was to enhance the creativity across the entire design system and within the teams. The CEO module of the control source was in charge of measuring and tracking the cognitive complexity of the integrated design system. The decomposition module of the control source was in charge of decomposing the problem after the generation stage of a given abstraction level was accomplished. The process of multi disciplinary team formation was based on the decomposition format of the problem, and SiFAs that were the elements of teams were grouped on this basis. The problem however may be decomposed in several modes and some of the literature on this issue was discussed in the previous section. Table 4 shows the decomposition modes that we have envisaged for IMMUNE. The decomposition module decides on one of these modes on the basis of problem connectivity for a given abstraction level.

This problem connectivity can be defined as the total number of edges in the parametric based DSM (and self map) of the problem divided by the total number of possible edges – that is, the number of edges of a complete graph with the same number of nodes. The total number of possible edges in a complete undirected graph with n nodes or vertices is

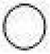
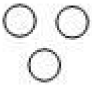
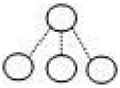
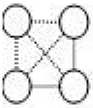
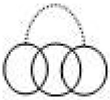
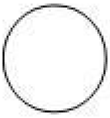
$$K = \binom{n}{2} = \frac{n \times (n-1)}{2} \tag{3}$$

For directed graphs, K is twice the amount presented in (3). If the self map of the simulated parameter based DSM has k connections (edges), we define the problem connectivity as:

$$p = \frac{k}{K} \tag{4}$$

It must be noted that the connectivity values presented in Table 4 are suggestions based on the experience gained from the experiments of authors with random graphs. More appropriate methods require more theoretical investigations into graph theory. We propose five modes of knowledge sharing and organizational structure that correspond to these decomposition modes: independent, integrative, multi-agent, collaborative and competitive:

Table 4 Modal decomposition of the problem at every abstraction layer based on the connectivity of the problem

Connectivity	Very Low (0-0.02)	Low (0.02-0.1)	Intermediate (0.1-0.2)	High (0.2-0.3)	Very High (0.3-1)
Possible or Best decomposition strategy	Full decomposition	Integrative clustering	Modular clustering	Overlap clustering	No decomposition
 Illustration: a subsystem					

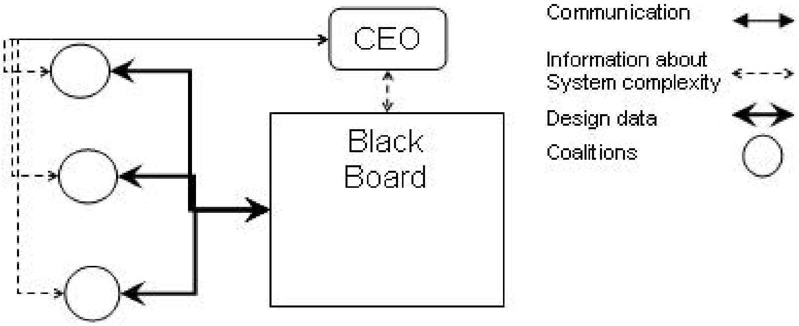


Fig. 19 Independent process mode: coalitions do not need to communicate

1. *Independent mode:* in this mode the decomposition agent has managed to fully decompose the problem; generally, very low self map connectivity leads to such situations. In this mode there is no collaboration between the coalitions as depicted in Figure 19 because when tasks are not interdependent, there is no need or reason to collaborate [43]. Consequently the need for radical innovation to integrate the system at the considered abstraction level would be minimal, and the process will be characterized by short lead times. However collaboration exists between the design agents *inside* the same coalition. The CEO monitors the cognitive complexity inside the coalitions using the system knowledge provided by the agents regarding the degree of interaction with other members of the coalition; this is the only control relationship in the system.

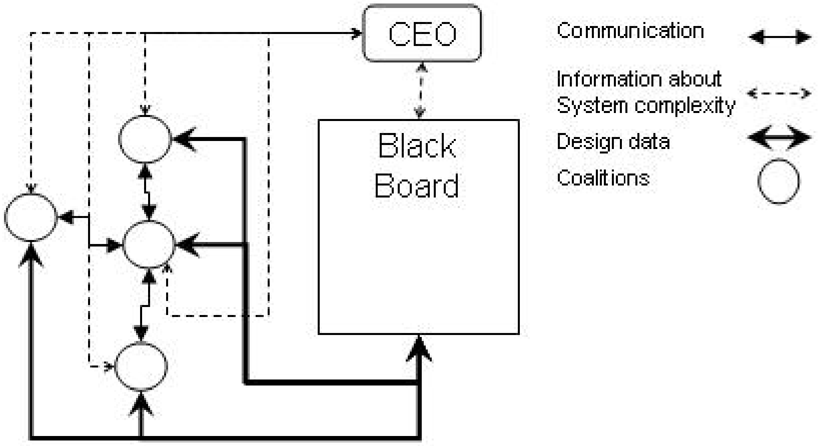


Fig. 20 Integrated process model. All coalitions exchange information with only one central coalition

2. *Integrative mode:* Integrative systems were reported and studied by Sosa et al. [63]; in these systems all subsystems are connected to a single subsystem, namely the integrative subsystem. The integrative decomposition is derived from the resequencing [63] of the parameter based DSM usually using integer programming [38]. Simple coordination of the design process makes this mode desirable, since all the coalitions have to coordinate their communications with only one integrative coalition. The corresponding organizational structure and integrative process mode is that illustrated in Figure 20. One drawback of this mode is that it might be hard for the design agents of the integrative coalition to maintain the cognitive complexity of the layer above the CEO-prescribed minimum cognitive complexity; which is to say one module must be able to reach a high cognitive complexity. In other words the coordination in between several coalitions through one coalition might not be feasible. As such integrative mode is advised for problems with low self connectivity.
3. *Modular decomposition and autonomy based process model:* Modular decomposition results in subsystems having significant connectivity to each other. In Efatmaneshnik and Reidsema [24] we have shown that the immune decomposition of systems into modular mode is less than that of integrative decomposition. As such modular decomposition is desirable for problems with intermediate self connectivity. The major criteria for clustering algorithms of modular decomposition are 1) to minimize the connectivity between the subsystems, 2) to maximize the connectivity of each subproblem. Both these criteria are met by minimization of the real complexity. Two significant issues that appear as decomposition constraints are the maximum number of subsystems and their minimum degree. The corresponding process model is depicted in Figure 21 and is referred to as an autonomy based process model. In this mode the agents explicitly act autonomously

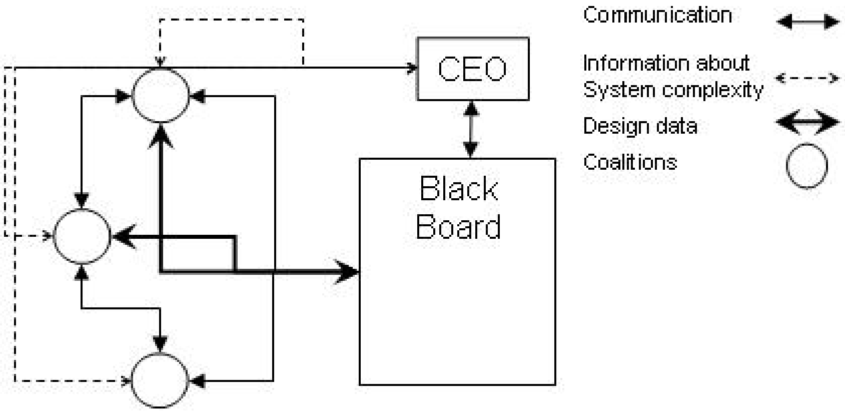


Fig. 21 Autonomy based process model

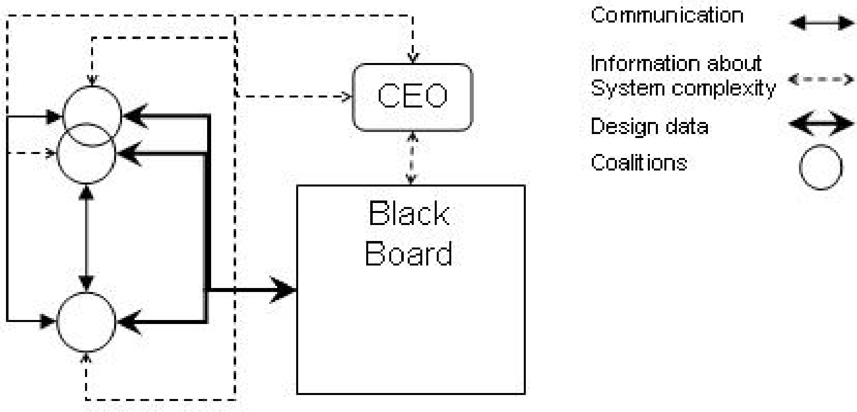


Fig. 22 Collaborative process mode

in their social behavior. The main characteristic of the autonomy process model is cooperation amongst the SiFAs inside and across the boundaries of coalitions.

4. *Overlap decomposition and collaborative process model:* In this mode subsystems are overlapped and they share some of the design variables with each other. As a result some of the coalitions explicitly share some of the agents, and there are some agents that have the membership of two or more coalitions (Figure 22). The real complexity is measured for overlap decompositions [24]. The main characteristic of this process model is the intense collaboration between coalitions that makes this mode an information and knowledge intensive process [36].
5. *No decomposition and competitive process model:* when the problems are very connected resulting in dense self maps, any kind of decomposition leads to large departures of the real complexity from the self complexity ($C_R >> C_S$). In this condition, decomposition may not be a solution to problem tractability.

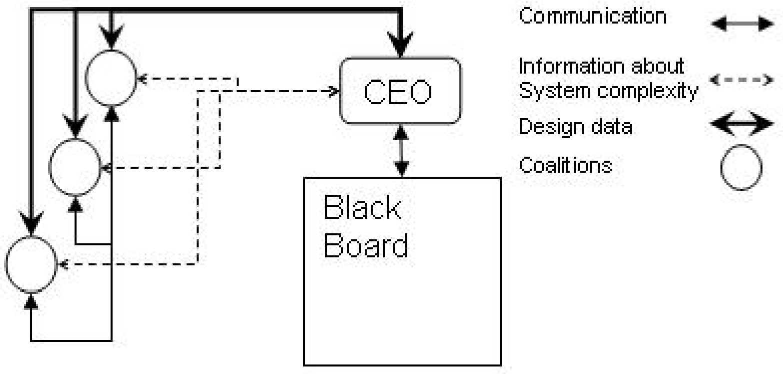


Fig. 23 Competitive process model

Bar-Yam [2] proposed the enlightened process model of problem solving for very connected problems. It was based on competition and cooperation between several design teams focused on solving the same problem. The main characteristic of this process model is competition between the design coalition (Figure 23). In the competitive mode the problem is not decomposed, and each coalition tackles the entire problem by itself. However, informal cooperation may exist between the coalitions, although there is no formal and explicit cooperation between the coalitions. The final solution is chosen from the submitted solutions of the coalition for the entire problem for a given abstraction level.

The quality of the solution is determined by the control source, based on the accuracy weights that the coalitions suggest for their solutions. In this mode the complexity is controlled only at the agents level (inside the coalitions) and the cognitive complexity arising from the informal cooperation of the coalitions is ignored.

6 Conclusion

Complex engineered systems can be conceived of as densely coupled systems. The design of such systems is a major challenge for engineering research. A new conceptual approach to concurrent parametric design of complex products was presented, representing it as a model of distributed computation and planning. IMMUNE is a multi-agent architecture that supports parametric decision making for complex products, employing a number of unique sub-solution methods in order to cope with complexity. These were:

Gradualism: Gradual implementation of the problem space through sequences of Generation, Decomposition, Distribution, and Integration of problems (GDDI cycle) [51]. Each sequence of the cycle represents the problem at a different abstraction level.

Monte Carlo Simulation and Global Entropy Based Correlation Coefficient [23]: are used to establish, early in the design phase, the self map of the system which shows the sensitivity of design and objective variables. This self map is represented as a weighted graph or parametric based design structure matrix [7]. A complexity measure is then applied to the graph to measure the complexity of the problem.

Immune Decomposition: The resulting complexity measure obtained through problem decomposition has been shown to be an indication of the system's real complexity which we have shown increases the overall complexity. We have defined this as a system's real complexity. Additional real complexity is the price paid for improving the tractability and manageability of a complex problem through decomposition. (No Free Lunch Theorem). A decomposition that has the least real complexity leads to a problem space being more robust and immune to chaotic behaviour.

Modal decomposition of problem space: A problem may be decomposed in several modes depending on the connectivity (or coupled-ness) of the problem variables at each level of abstraction. These decomposition modes are described as being analogous to the growing connectivity of the problem and are defined as: 1) Full decomposition, all subsystems (subproblems) are independent for least connected systems. 2) Integrative or coordination based decomposition; where one subsystem (named integrative subsystem) is connected to all other independent subsystems that are independent. (3) Modular or multi-agent decomposition, where all subsystems or some of them are connected. 4) Overlap decomposition [24], which is similar to multi-agent decomposition with the exception that some of the subsystems are overlapped indicating shared design and objective variables. 5) No decomposition [3] for densely connected systems.

Adaptive Structuration [19]*:* The proposed architecture is capable of planning decisions in a metamorphose environment [47] for each of the five mentioned decomposition modes at every GDDI cycle. Design agents are clustered within each GDDI cycle as virtual teams or coalitions of agents whose structure mimics the structure of the problem. Subsequently, and correspondingly to the five modes of decomposition, the IMMUNE architecture is capable of employing five modes of design: 1) Independent mode that is fully concurrent problem solving. 2) Integrative mode which is coordination based problem solving. 3) Autonomy based problem solving that is cooperative and on the basis of cooperation of several coalitions of agents. 4) Collaborative problem solving where some of the coalitions of agents are semi merged and overlapped [36]. 5) Competitive problem solving on basis of Enlightened Engineering [2] in which several independent coalitions of agents competing to solve the same problem.

Global blackboard data base: Adaptive Structuration is accomplished using a global blackboard containing the current state of the design at all abstraction levels [5]. The control source decides on the decomposition mode based on the connectivity of the problem and then decomposes it on the basis of minimum real complexity.

Monitoring of the design process complexity using complexity bounds: Since the relationships between system parameters in a complex system are often nonlinear, the development and integration of such systems is often obscure. Nonlinear systems may exhibit chaotic properties that are non-integrative systems. Cognitive complexity is the ability of a person or an organization to integrate a system [42]. In order to integrate and manage a complex system the problem solving central management unit requires a complexity that is more, or at least equal, to the complexity of the problem [2] and that is the cognitive complexity of the organization. In this paper the minimum and maximum cognitive complexity bounds were measured from the simulated parameter based design structure matrix by the CEO module of the blackboard control source. The collective cognitive complexity of a product development organization is tied to the extent its units are connected [42], and therefore can be measured as a function of the amount of information exchange between the design agents inside one coalition and in between the coalitions. Correspondingly, the CEO monitors the cognitive complexity at two hierarchical levels: 1) low level and inside each coalition 2) high level that is the entire federation of coalitions in any abstraction level. The CEO informs all the design agents of the amount of cognitive complexity of their coalition and the federation. The COPE module of the design agents are then in charge of maintaining the cognitive complexity of the coalitions and the federation above the announced (by CEO) minimum away from the maximum bound. The COPE module, decides on the high level interactions mode (passive or proactive-social) by using the conflict resolution strategies that are passive like constraint relaxation or proactive such as active negotiation.

By utilizing the aforementioned tools the fragility of the development process of a complex system may be dealt with. The presented architecture is IMMUNE against sudden failure in meeting the top level organization objectives including cost, lead time and the quality of the product. It is often argued that complex systems are robust yet in the presence of uncertainties they become fragile; this strange behaviour is related to the chaotic and sensitive characteristic of complex systems. In the domain of sustainability of the organizations that design complex products this means that the top level goals may often be robustly met, however, sudden and large departures from those goals may seem inevitable. To immunize against this fragility the proposed system advocates coherency in collaboration. That is, the locally aware design agents (aware of their local tasks) maintain the global coherency, harmony and order through their COPE module by making the agents' social behaviour subservient to the information the system's cognitive complexity received from the CEO module.

References

1. Bar-Yam, Y.: When Systems Engineering Fails — Toward Complex Systems Engineering. In: Proceedings of International Conference on Systems, Man & Cybernetics, pp. 2021–2028. IEEE Press, Piscataway (2003)
2. Bar-Yam, Y.: Making Things Work: Solving Complex Problems in a Complex World. NECSI Knowledge Press (2004)
3. Bar-Yam, Y.: Engineering Complex Systems: Multiscale Analysis and Evolutionary Engineering. In: Complex Engineered Systems, pp. 22–39. Springer, Heidelberg (2007)

4. Bayrak, C., Tanik, M.M.: A Process Oriented Monitoring Framework. Systems Integration 8, 53–82 (1998)
5. Brown, D.C., Dunskus, B., Grecu, D.L., et al.: SINE: Support for single function agents. In: Applications of AI in Engineering, Udine, Italy (1995)
6. Browning, T.R.: Designing system development projects for organizational integration. Systems Engineering 2, 217–225 (1999)
7. Browning, T.R.: Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. IEEE Transactions on Engineering Management 48, 292–306 (2001)
8. Carver, N., Lesser, V.: The Evolution of Blackboard Control Architectures. Expert Systems with Applications 7, 1–30 (1994)
9. Chen, L., Li, S.: Concurrent Parametric Design Using a Multifunctional Team Approach. In: Design Engineering Technical Conferences DETC 2001, Pittsburgh, Pennsylvania (2001)
10. Chiva-Gomez, R.: Repercussions of complex adaptive systems on product design management. Technovation 24, 707–711 (2004)
11. Cisse, A., Ndiaye, S., Link-Pezet, J.: Process Oriented Cooperative Work: an Emergent Framework. In: IEEE Symposium and Workshop on Engineering of Computer Based Systems, Friedrichshafen, Germany, pp. 342–347 (1996)
12. Cohen, I.: Real and Artificial Immune Systems: Computing the State of the Body. Imm. Rev. 7, 569–574 (2007)
13. Corkill, D.D.: Blackboard Systems. AI Expert 6, 40–47 (1991)
14. Corkill, D.D.: Collaborating Software: Blackboard and Multi-Agent Systems & the Future. In: International Lisp Conference, New York (2003)
15. Craig, I.D.: Formal Techniques in the Development of Blackboard Systems. In: Research Report Coventry, Department of Computer Science, University of Warwick, UK (1993)
16. Dasgupta, D.: An Artificial Immune System as a Multi-Agent Decision Support System. In: IEEE International Conference on Systems, Man and Cybernetics (SMC), San Diego, California, vol. 4, pp. 3816–3820 (1998)
17. Dembski, W.: No Free Lunch: Why Specified Complexity Cannot Be Purchased without Intelligence. Rowman & Littlefield Publishers, Inc (2002)
18. DeSanctis, G., Monge, P.: Communication processes for virtual organizations. Organization Science 10, 693–703 (1999)
19. Desanctis, G., Poole, M.S.: Capturing the Complexity in Advanced Technology Use: Adaptive Structuration Theory. Organization Science 5, 121–147 (1994)
20. Dissanayake, K., Takahashi, M.: The Construction of Organizational Structure: Connections with Autopoietic Systems Theory. Contemporary Management Research 2, 105–116 (2006)
21. Druzdzel, M.J., Flynn, R.R.: Decision Support Systems. In: Kent, A. (ed.) Encyclopedia of Library and Information Science, vol. 67(30), pp. 120–133. Marcel Dekker, Inc., New York (2000)
22. Dunskus, B.V.: Single Function Agents and Their Negotiation Behavior in Expert Systems. Worcester Polytechnic Institute, Worcester (1994)
23. Efatmaneshnik, M., Reidsema, C.A.: Immunity as a Design Decision Making Paradigm for Complex Systems: a Robustness Approach. Cybernetics and Systems 38(8), 759–780 (2007a)
24. Efatmaneshnik, M., Reidsema, C.A.: Immunity and Information Sensitivity of Complex Product Design Process in Overlap Decomposition. In: Minai, A., Braha, D., Bar-Yam, Y. (eds.) Proceedings of 7^{th} ICCS, Boston, MA (2007b)

25. Formica, A.: Strategic Multiscale A New Frontier for R&D and Engineering. Ontonix, Turin (2007), `http://www.ontonix.com/index.php?page=download&CID=36`
26. Fruchter, R., Clayton, M.J., Krawinkler, H., et al.: Interdisciplinary communication medium for collaborative conceptual building design. Advances in Engineering Software 25, 89–101 (1996)
27. Fyfe, C., Jain, L.: Teams of intelligent agents which learn using artificial immune systems. Journal of Network and Computer Applications 29, 147–159 (2006)
28. Gero, J.S.: Creativity, emergence and evolution in design. Knowledge-Based Systems 9, 435–448 (1996)
29. Ghanea-Hercock, R.: Survival in cyberspace. Information Security 12, 200–208 (2007)
30. Goel, S., Gangolly, J.: On decision support for distributed systems protection: A perspective based on the human immune response system and epidemiology. International Journal of Information Management 27, 266–278 (2007)
31. Gulati, R.K., Eppinger, S.D.: The coupling of product architecture and organizational structure decisions. MIT, Cambridge (1996)
32. Henderson, R.M., Clark, K.B.: Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. Administrative Science Quarterly 35 (1990)
33. Hinds, P., McGrath, C.: Structures that work: social structure, work structure and coordination ease in geographically distributed teams. In: Proceedings of 2006 the 20th anniversary conference on Computer supported cooperative work (CSCW 2006), Banff, Alberta, Canada, November 04-08, pp. 343–352 (2006)
34. Hobday, M., Rush, H., Tidd, J.: Innovation in complex products and system. Research Policy 29, 793–804 (2000)
35. Kan, J., Gero, J.: Can entropy represent design richness in team designing? In: Bhatt, A. (ed.) CAADRIA 2005, New Delhi, pp. 451–457 (2005)
36. Klein, M., Sayama, H., Faratin, P., et al.: The Dynamics of Collaborative Design: Insights from Complex Systems and Negotiation Research. Concurrent Engineering Research & Applications (2003)
37. Kratzer, J., Leenders, R.T.A.J., Engelen, J.M.L.V.: A delicate managerial challenge: how cooperation and integration affect the performance of NPD teams. Team Performance Management 10, 20–25 (2004)
38. Kusiak, A.: Engineering Design: Products, Processes, and Systems. Academic Press, London (1999)
39. Lander, S.E., Staley, S.M., Corkill, D.D.: Designing Integrated Engineering Environments: Blackboard-Based Integration of Design and Analysis Tools. Concurrent Engineering: Research and Applications 4, 59–72 (1996)
40. Larson, J.R.: Deep Diversity and Strong Synergy: Modeling the Impact of Variability in Members' Problem-Solving Strategies on Group Problem-Solving Performance. Small Group Research 38, 413–436 (2007)
41. Lau, H., Wong, V.: Immunologic Responses Manipulation of AIS Agents. In: Nicosia, G., Cutello, V., Bentley, P.J., Timmis, J. (eds.) ICARIS 2004. LNCS, vol. 3239, pp. 65–79. Springer, Heidelberg (2004)
42. Lee, J., Truex, D.P.: Cognitive Complexity and Methodical Training: Enhancing or Suppressing Creativity. In: 33rd Hawaii International Conference on System Sciences (2000)
43. Leenders, R.T.A.J., Kratzer, J., Hollander, J., et al.: Virtuality, Communication, and New Product team Creativity: a Social Network Perspective. Engineering and Technology Management, 69–92 (2003)

44. Lissack, M.: Complexity: the Science, its Vocabulary, and its Relation to Organizations. Emergence 1(1), 110–126 (1999)
45. Marczyk, J.: Principles of Simulation Based Computer Aided Engineering. FIM Publications (1999); BarcelonaSycara, K.: Multiagent Systems. AI Magazine 19(2), 79–93 (1998)
46. Marczyk, J., Deshpande, B.: Measuring and Tracking Complexity in Science. In: Minai, A., Braha, D., Bar-Yam, Y. (eds.) 6^{th} ICCS, Boston, MA (2006)
47. Maturana, F., Shen, W., Norrie, D.H.: MetaMorph: an Adaptive Agent-Based Achitecture for Intelligent Manufacturing. International Journal of Production Research 37, 2159–2173 (1999)
48. Monceyron, E., Barthes, J.P.: Architecture for ICAD Systems: an Example from Harbor Design. Sience et Techniques de la Conception 1, 49–68 (1992)
49. Prasad, B.: Concurrent Engineering Fundamentals. Integrated Product Development, vol. II. Prentice Hall, Englewood Cliffs (1996)
50. Quadrel, R.W., Woodbury, R.F., Fenves, S.J., et al.: Controlling asynchronous team design environments by simulated annealing. Research in Engineering Design 5, 88–104 (1993)
51. Reidsema, C., Szczerbicki, E.: Towards a System for Design Planning in a Concurrent Engineering Environment. International Journal of Systems Analysis, Modeling, and Simulation 29, 301–320 (1997)
52. Reidsema, C., Szczerbicki, E.: Blackboard Approach in Design Planning for Concurrent Engineering Environment. Cybernetics and Systems 29, 729–750 (1998)
53. Reidsema, C., Szczerbicki, E.: A Blackboard Database Model of the Design Planning Process in Concurrent Engineering. Cybernetics and Systems 32, 755–774 (2001)
54. Reidsema, C., Szczerbicki, E.: Review of Intelligent Software Architectures for the Development of An Intelligent Decision Support System for Design Process Planning in Concurrent Engineering. Cybernetics and Systems 33, 629–658 (2002)
55. Rosenman, M., Wang, F.: CADOM: A Component Agent-based Design-Oriented Model for Collaborative Design. Research in Engineering Design 11, 193–205 (1999)
56. Saad, M., Maher, M.L.: Shared understanding in computer-supported collaborative design. Computer-Aided Design 28, 183–192 (1996)
57. Shen, W., Barthès, J.P.: An Experimental Multi-Agent Environment for Engineering Design. International Journal of Cooperative Information Systems 5, 131–151 (1996)
58. Shen, W., Norrie, D.H.: A Hybrid Agent-Oriented Infrastructure for Modeling Manufacturing Enterprises. In: KAW 1998, Banff, Canada, pp. 117–128 (1998)
59. Shen, W., Norrie, D.H., Barthès, J.-P.: Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing. CRC Press, Boca Raton (2001)
60. Sinha, R., Liang, V.C., Paredis, C.J.J., et al.: Modeling and Simulation Methods for Design of Engineering Systems. Computing and Information Science in Engineering 1, 84–91 (2001)
61. Sosa, R., Gero, J.: Diffusion of Creative Design: Gate keeping Effects. International Journal of Architectural Computing 2, 518–531 (2004)
62. Sosa, R., Gero, J.: A Computational Study of Creativity in Design. AIEDAM 19, 229–244 (2005)
63. Sosa, M.E., Eppinger, S.D., Rowles, C.M.: Designing Modular and Integrative Systems. In: DETC 2000 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Baltimore, Maryland (2000)
64. Stacey, R.D.: The Science of Complexity: An Alternative Perspective for Strategic Change Processes. Strategic Management Journal 16, 477–495 (1995)
65. Sycara, K.: Multiagent Systems. AI Magazine, 79–93 (1998)

66. Timmis, J., Andrews, P., Owens, N., et al.: An Interdisciplinary Perspective on Artificial Immune Systems. Evolutionary Intelligence 1, 5–26 (2008)
67. Tomiyama, T., Umeda, Y., Ishii, M., et al.: Knowledge systematization for a knowledge intensive engineering framework. In: Tomiyama, T., Mantyla, M., Finger, S. (eds.) Knowledge Intensive CAD-1, pp. 55–80. Chapman & Hall, Boca Raton (1995)
68. Turban, E.: Decision support and expert systems: management support systems. Prentice Hall, Englewood Cliffs (1995)
69. Wolf, T.D., Holvoet, T.: Towards a Methodology for Engineering Self-Organising Emergent Systems. In: Self-Organization and Autonomic Informatics, Glasgow, UK, pp. 18–34 (2005)
70. Wong, A., Sriram, D.: SHARED: An information model for cooperative product development. Research in Engineering Design 5, 21–39 (1993)
71. Zdrahal, Z., Motta, E.: Case-Based Problem Solving Methods for Parametric Design Tasks. In: Proceedings of the Third European Workshop on Advances in Case-Based Reasoning (EWCBR 1996), pp. 473–486. Springer, London (1996)
72. Zhang, C.: Cooperation under uncertainty in distributed expert systems. Artificial Intelligence 56, 21–69 (1992)

Bayesian Learning for Cooperation in Multi-Agent Systems

Mair Allen-Williams and Nicholas R. Jennings

Abstract. Multi-agent systems draw together a number of significant trends in modern technology: ubiquity, decentralisation, openness, dynamism and uncertainty. As work in these fields develops, such systems face increasing challenges. Two particular challenges are decision making in uncertain and partially-observable environments, and coordination with other agents in such environments. Although uncertainty and coordination have been tackled as separate problems, formal models for an integrated approach are typically restricted to simple classes of problem and are not scalable to problems with many agents and millions of states. We improve on these approaches by extending a principled Bayesian model into more challenging domains, using heuristics and exploiting domain knowledge in order to make approximate solutions tractable. We show the effectiveness of our approach applied to an ambulance coordination problem inspired by the Robocup Rescue system.

1 Introduction

As computing power and ubiquity increase, the use of multi-agent technology in large distributed systems is becoming more widespread. For example, sensors are often now included in new buildings or vehicles. When these sensors are able to sense intelligently and communicate with one another, they form a multi-agent system. Mobile sensors may be able to make inferences about a scenario such as a terrorist attack or a flood, and provide human teams with uncertainty estimates and suggest actions. In situations where a human would be at some risk, intelligent communicating machines may be deployed—for example, thousands of UAVs (unmanned aerial vehicles) can collaborate to search over a wide area [32], or robots may also act as intelligent agents, aiding or replacing humans to perform a coordinated search of a burning building [33]. Consequently, as such systems develop, the scalability of complex interacting systems becomes increasingly important.

Mair Allen-Williams and Nicholas R. Jennings
School of Electronics and Computer Science, University of Southampton, SO17 1BJ, UK
e-mail: mhaw05r,nrj@ecs.soton.ac.uk

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 321–360.
springerlink.com

In more detail, an autonomous agent is an entity which makes local perceptions within an environment and processes these perceptions in order to decide how to act on that environment, based on some internal goals. When many such agents are acting within the same environment then the actions of one can affect the perceptions of others. This is the essence of a multi-agent system and the reason why cooperation is essential to its effective operation.

In order to provide a focus, and to motivate this work, we will consider the disaster response domain as an example multi-agent system. Disaster scenarios form rich grounds for multi-agent distributed problem solving, allowing us to explore several features of complex multi-agent problems. While there are many characteristics which may be present in disaster scenarios, we will find that there are two common themes: *uncertainty*, and *coordination*.

The first of these, uncertainty, may concern the environment ("What's going on?") and the agent's position in the environment ("Where am I?"); it may be about any other agents which might exist in the environment ("Who else is around? Where are they?") and their behaviour ("What are they going to do?"). In these uncertain situations, each agent must do some form of discovery to determine the essential characteristics of the scenario, including the agent's collaborators, before and alongside directly working to achieve its goals. This discovery phase in a multi-agent system is tightly linked with the presence of other agents in the system. As well as determining which other agents are present, agents may be able cooperate to search over different regions, sharing information with each other as appropriate.

In addition to explicitly sharing information, observing the behaviour of the other agents allows an autonomous agent to make inferences about the system. For example, in a scenario involving a burning building, a rational agent will not enter the building (although a specially designed robot or one which believes itself to be expendable may). Beyond discovery, there will continue to be interaction between the agents in a multi-agent system, whether explicit via communications and negotiations, or implicit through activity. Moreover, achieving some subgoals may involve a collaboration between several agents, as in a rescue operation where two ambulance members are required to carry a stretcher, or a driving team with a navigator and a driver.

Now, this general problem of taking others into account, *coordination*, is the second key issue we have identified for multi-agent systems. In uncertain or open systems, fixed protocols for coordination must function against a background where agents are not fully aware of the situation; that is, their environment, the resources available to them, or the behaviour of the other agents. The negotiation of coordinated behaviour in such systems is intertwined with the discovery phase, as agents interact with one another, perhaps cooperating to determine properties of the situation.

In order to achieve such a comprehensive model for planning and acting, we have built on existing techniques for cooperative decision making under uncertainty. Now, the inherent dynamism in many of these problems calls for timely online responses, rather than offline computation of strategies. For partially observable multi-agent problems, recent work has advanced the state of the art for finding offline solutions in systems of stationary agents, with solutions in systems containing at least fifteen agents (where previously the maximum was five or six) [23]. Building on this, we describe a related *online* approach which is suitable for complex dynamic systems with mobile agents, and also scales to tens of agents.

Our algorithm explicitly models other agents, demonstrating a principled approach to cooperative behaviour in uncertain and partially-observable multi-agent systems. Through empirical evaluation, we show that our approach, using learned finite state machines to approximate the behaviour of others, is more efficient than other principled approaches [14] and more effective than a state-of-the-art hand-written strategy for the same scenario. In practical terms, this algorithm could be directly deployed in systems such as collaborative mobile sensor networks or for UAVs sharing a distributed search. In more human-led systems, such as a disaster response, we anticipate that communicating sensors in both fixed networks (for example, in buildings) and hand-held applications could suggest actions or action sequences to human participants. While in the future we expect that robot teams will be able to become increasingly autonomous with valuable contributions to such scenarios, in this chapter we think it best to consider our 'agents' to be computer-advised humans.

Over the next sections, we provide a detailed background to our work (Section 2) and motivate a cooperative approach using finite state machines to model agent behaviour. In Sections 3 and 4 we describe this approach in detail. In order to validate our model we test it on a problem taken from the disaster response domain. Section 5 describes this problem and Section 6 compares the performance of our algorithm with previous approaches to partially observable uncertain systems. In Section 7 we conclude and describe directions for future work.

2 Background

In this Section we introduce the ideas which we will use in our algorithm, explaining the way in which the multi-agent approach to partially-observable systems is developed from single-agent decision theory, and justifying the decisions we have made at each step. First, however, we introduce the disaster response domain as a motivation for this work and identify its key characteristics.

2.1 The Disaster Response Domain

We ground our work in the disaster response domain. After a disaster such as an earthquake or a flood, the immediate situation and its environmental properties are typically unknown to the rescue teams, and the complete situation cannot be atomically observed by a single agent. Rescue teams may come from different regions but all must collaborate to search the area and rescue any disaster victims in a timely fashion. However, communication lines may be unavailable or restricted so that this collaboration is necessarily implicit or based on short one-way communications.

In more detail, we find that taking disaster response as our focus domain drives a particular interest in collaborative multi-agent domains which include the following properties:

Decentralisation: In these large and dynamic systems, providing a central controller is likely to be infeasible. Firstly, there are unlikely to be sufficient resources to allow communications between one central controller and every other node. Secondly, one central controller is almost certainly not going to be able to obtain a complete view of the system, and the potentially rapid changes as agents enter and leave the system would be difficult to track.

Dynamism: Realistic systems are rarely static. For example, in disaster recovery agents must adapt to changing weather conditions, any aftershocks, and unexpected events such as building collapse or fires. When taken together, this can lead to a very dynamic environment.

Partial observability: Along with decentralisation, it is likely that no one agent is able to see the complete system all the time. Although communication between agents may extend a particular agent's view of the system, the agent must continually make judgements based on an incomplete view.

Bandwidth-limited: Limited communication is a characteristic common to disaster scenarios—for example, mobile phone networks often become jammed [24], or time constraints can limit opportunities for communication. Thus, agents may be able to exchange some information, but both time and bandwidth restrictions will limit these exchanges.

Openness: The rescue agents are likely to be entering or leaving the disaster scene throughout the rescue operation. Agents may be harmed at the scene and thenceforth be out of action, while new agents may arrive late. A collaborative model in a disaster response scenario must therefore be able to adapt to the continual arrival and loss of agents.

Example 1 describes an earthquake scenario having many of these features. Keeping these driving forces in mind, we begin with a description of the most straightforward of this class of dynamic problems, the single agent observable Markov Decision Process (MDP) (Section 2.2.1). Building on the single agent MDP, we will generalise to partially observable (Section 2.2.4) and multi-agent environments (Section 2.3), discussing means of coordination among agents. We will not discuss open systems in detail until the final Section (Section 7).

2.2 Single Agent Decision Making

We begin with an introduction to decision making processes for a single agent acting in a dynamic environment, going on to explain how the agent can learn about uncertain environments via reinforcement learning, and introducing Bayesian reinforcement learning. We then extend these decision making techniques into partially observable environments, again using Bayesian techniques.

Example 1. Earthquake scenario

An earthquake has damaged the small town of Tameugny (Figure 1), including its hospital and ambulance fleet. Buildings are still collapsing and there may be aftershocks (*dynamism*). Ambulance teams from nearby towns converge on Tameugny, travelling through the damaged streets searching for hurt victims among the rubble (*partial observability*). Although the ambulance dispatch stations are able to communicate with one another, once the ambulances are on the road to Tameugny, they find the communications networks are blocked (*bandwidth limitations*). They must therefore make decisions independently (*decentralisation*), leaning out of their windows to warn other ambulance drivers about damaged roads, exploring parts of town not yet marked by emergency services' red and white tape, or going to the aid of ambulance teams working in particularly damaged areas, such as a collapsed office building. They also need to learn about the capabilities of ambulances from other towns, who may be equipped differently or even have different goals—one apparent ambulance turns out to be a concealed news team. As the ambulances come and go, the system is *open*.

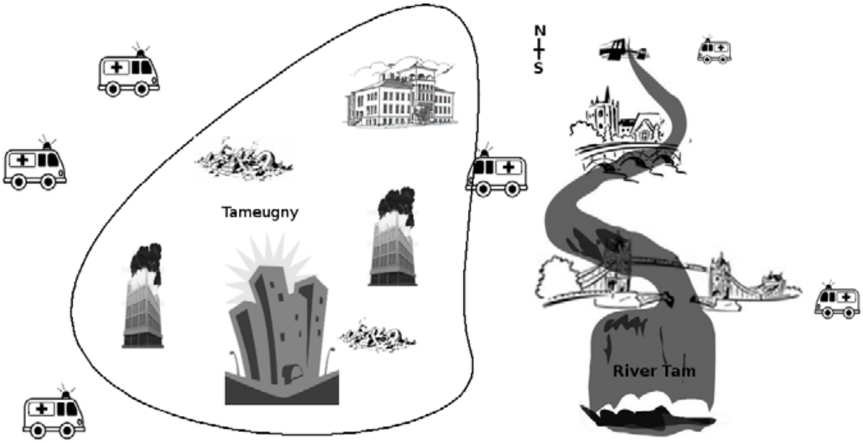


Fig. 1 Tameugny, after an earthquake

2.2.1 Markov Decision Processes

In this most basic model, the agent perceives the state of the world through its sensory inputs, and decides on its immediate action based on this state. Following the agent's action, the world *transitions* into a new state, and the agent may receive some *reward*. This model forms the basis of Markov decision theory [36]. The fundamental feature of this theoretical model is the assumption that the immediate next state is dependent only on the previous state and choice of action—this is the *Markov property*. Although the Markov property may not fully hold, it is often a sufficiently good approximation, and techniques which use this theory can get good results. This is demonstrated by many practical examples [18] [35] [2]. With the Markov assumption, if the models describing the transition and reward probabilities are completely known to the agent then the system can be solved, using a pair of recursive equations [36] which determine the optimal action from each world state. These are the

Bellman equations. For large systems, there are efficient ways of approximating these solutions—we do not go into these here, as we will not be dealing with known MDPs, but refer the interested reader to [36], Chapter 9.

2.2.2 Reinforcement Learning

Example 2. Rescue worker in the old people's home

A lone rescue worker searches Tameugny's old people's home after the earthquake. As she works her way up the building, she takes increasing care how she treads, not knowing what structural damage the earthquake may have caused—the environmental dynamics are *uncertain*. Some parts of the building were more heavily populated than others—the dining area was full of elderly people and waiters; most of the bedroom wings are almost empty, but the Violet Wing was being cleaned by a team of a dozen cleaners. The rescue worker does not initially know how the building was laid out or which areas were most crowded, and must discover this as she makes her way through the building.

When there is uncertainty about the aforementioned models, as in Example 2, the agent can learn the optimal actions through experimentation. To this end, *reinforcement learning* techniques, such as Q-learning, TD(λ) and SARSA [36], provide techniques for the agent to do this. There are two types of learning: model-based and model-free. In the former, the agent aims to learn the system model, in this case the underlying MDP, and then solve that model (using the Bellman equations, as above) to decide an action. In the latter, the agent learns a direct mapping from the state to the optimal action. Model-free learning typically involves simple updates at each step and is consequently often more efficient for one-off problems. However, in comparison, model-based methods can be used to carry out many simulation steps alongside each real-time step, taking advantage of otherwise idle cpu cycles in relatively slow-progressing problems. Another advantage of model-based methods is the ability to bias the system towards a particular real model, using domain knowledge. Models or parts of models can also be re-used in different problems. Given this, we will focus on model-based methods particularly because of these two properties: in scenarios such as disaster response we *will* have initial beliefs about the system based on the domain or similar disasters and would like to incorporate those beliefs into our solutions.

2.2.3 Bayesian Reinforcement Learning

In particular, we focus on Bayesian model-based learning. By comparison to most model-based learning methods, which maintain a point estimate of the models, a Bayesian learning method will maintain a probability distribution over all possible models, in the form of a *belief state*. This provides a principled solution to the *exploration-exploitation* problem (Example 3): the decision an agent has to make between taking the action it currently believes to be optimal, and taking an exploratory action. In general, the more certain the agent is about its current model, the more likely it should be to take the currently optimal action. The Bayesian model pins this intuition down precisely.

Example 3. Exploration-exploitation in the Tameugny earthquake

The river Tam runs to the east of Tameugny. As ambulances rush in to the rescue from the east, they find that the earthquake has also destroyed several of the bridges across the river. An ambulance arriving at the riverside early after the disaster is able to learn over the radio that there is a bridge still standing two miles downriver. However, there is no data about the bridges upriver. The ambulance driver knows that there is a bridge only half a mile away, if it is still standing, and another a mile and half away, but then no more bridges for five miles. The decision the ambulance driver must make about whether to travel in the uncertain direction, or head straight for the bridge which is known to be standing, is an example of an *exploration-exploitation* problem.

2.2.4 Partially Observable Markov Decision Processes

Now, reinforcement learning, combined with the Bellman equations, will allow a single agent to solve any observable MDP which comes its way. However, although MDP models will form the basis of our environment, in large or complex scenarios it is common for an agent to make local observations which allow it to form inferences about the current state (Example 4), without observing the complete state directly (although in multi-agent systems, local observations may be augmented with communicated information). When the underlying process of moving from global state to global state is still (assumed to be) Markov, the scenario is described as a *partially observable Markov decision process*, or POMDP, and there are a host of POMDP-solution techniques.

Example 4. Partial observability in the Tameugny earthquake

Two Tameugny ambulances which survived the earthquake immediately swing into action. However, beyond the strength that they felt the earthquake to be, they have no idea of the scale or the detail of the situation. Elsewhere, as an office worker runs from a crumbling building, an approaching ambulance calls out to ask how many people were in the building—the answer (an estimate) is information which will remain local to that ambulance until much later. In other parts of town, other ambulances will have their own local information. However, the big picture will not be completed until much later on, if at all.

In particular, when the underlying environmental model is known, the POMDP can be converted to a continuous Markov decision process by defining a *belief state* as a probability distribution over states. The resulting continuous MDP, from belief state to belief state, can be solved using exact algorithms [7] or using approximations to make computation easier [3], [20]. If the underlying model is not known, learning techniques must be used to refine a solution as the agent explores the system. Model-free approaches, such as [1], have had some success in using learning techniques to solve POMDPs. However, as discussed, we believe that model-based approaches may again have benefits—for example, [34] demonstrates a model-based algorithm which uses variable length suffix trees to address the fact that even if state transitions are Markov, the observable process may not be. However, existing approaches rely on a number of approximations and assumptions about the state space, hence are not entirely satisfactory. A principled approach may be to extend the Bayesian model described previously into partially observable domains [30].

2.3 Coordinated Decision Making

Above, we have discussed agents reasoning about their environments. However, as well as reasoning about their environment, agents in a multi-agent system will be interacting with each other (Example 5). This interaction can be modelled by defining a (hyper)sphere of influence for each agent within the environment. Overlapping spheres of influence indicate interactions between agents [41]. A model of how different spheres interact will form a part of the agent's model of the system, as will models of the behaviour of the other agents. Making decisions in the context of these other agents is the fundamental principle of coordination [12]. Clearly, this is a central part of a reasoning agent in a multi-agent system. Thus, in the following sections we expand on how agents can reason about the behaviour of others and incorporate that reasoning into their own behaviour.

Example 5. Ambulance traffic at the the Tameugny earthquake

Consider again the ambulance driver arriving at the River Tam after the Tameugny earthquake. If he is the only ambulance approaching the scene, he may choose not to take the risk of having to travel many miles upriver, and head straight for the bridge which is known to be standing. However, if he knows that there is a fleet of ambulances following him, he may choose to head upriver so that he can (subject to communication networks functioning) send back data about the status of the bridges to later ambulances, enabling them to update their model without the travel costs. He might also consider that if all the ambulances were to head for the one bridge, a traffic jam would form there, perhaps wasting precious time.

Perhaps the simplest example: agents functioning in uncertain worlds among other agents may include others' behaviour in the Markov state transition model they develop. However, by doing this they may form inaccurate assumptions about the world, as agents adapt their behaviour to one another. Consequently, maintaining models of the world and of other agents separately provides greater flexibility and may enable the agent to reuse a world model as agents come and go, or reuse models built for known agents in fresh scenarios. Below, we outline three common ways in which agents may develop and use models of the world to coordinate.

2.3.1 Coordination Mechanisms

Three, potentially overlapping, coordination mechanisms are identified by [4]: conventions, communication, and learning (Example 6). Firstly, **conventions** are typically the simplest form of coordination. In a convention-based coordination system, there are a number of assumed "social rules" describing ways for agents to interact when they are aware of other agents. Coordination by convention is typically simple, scalable and requires no setup time [16]. However, it is inflexible, and relies on all participants knowing the conventions and complying with them. More complex models using conventions include role-based structures [37] and self-organising structures [39]. Secondly, **communication** is used for coordination in many kinds of system. Coordination through communication has a small setup time and some bandwidth costs. In most large systems there will be some form of communication in order to share information between agents; it will be impossible for any one agent to sense all the information it needs to function effectively in context [13]. However, we expect to make limited use of communication beyond information-sharing, as the bandwidth and timeliness constraints will typically preclude it. Finally, it is possible to extend single agent **learning** into the multi-agent domain. The uncertainties of our target domain make learning techniques a natural approach to problems within this domain. Learning techniques enable agents to evolve coordinated polices within uncertain state spaces, either with a group of learners exploring the space and converging towards an equilibrium (as in [10] and [22]), or by one agent explicitly learning about the behaviour of others in order to adapt its own appropriately [8].

Example 6. Coordination mechanism examples

Consider the ambulance drivers approaching the Tameugny earthquake. They can use all three of the above coordination mechanisms:

- By **convention**, they will drive on the left hand side of the road; they will use sirens to indicate their approach; they will rescue the elderly, the mothers and the children first.
- They will **communicate** with the other ambulance drivers, calling things like "road's gone up there", "I think someone needs to check out the North-East of the town", "We need three more people to help lift here". By convention, sirens also communicate with others.
- As they work with other ambulance teams, they will **learn** which teams need the most help, which areas of the city have been searched, and how a particular team tends to operate (for example, whether they use *one-two-**lift***, or *one-two-three-**lift***)

2.3.2 Game Theory

Distinct from the three approaches to coordination identified above, another research domain which investigates coordination from a theoretical angle is **game theory** [21]. In game-theoretic formulations, agents model the scenario as a game and try and derive, either through exact evaluation or through learning, a *best response* to the strategies of the other players in the game (Example 7). If all the players iteratively keep playing best responses, and if strategies are *mixed* (stochastic) the play will converge to a (mixed) equilibrium, in which every player's strategy is a best response to every other player. One of the challenges of game theory is to direct the play so that convergence is not just to any equilibrium but to an optimal one [10].

Within the domain of game theory, the form of multi-agent learning in which the agents maintain explicit models of the other agents is described as learning in stochastic games. One effective approach to extending single-agent reinforcement learning into this setting is the *win-or-learn-fast* (WoLF) approach: an agent's learning rate is adjusted according to its current performance, without explicitly modelling the other agents [5]. However, WoLF techniques can be improved upon by using a Bayesian model in which agents maintain beliefs about the behaviour of the other agents, as well as a probability distribution over world models [8]. The need for heuristically determined learning rates is then eliminated, while prior information about agents can be incorporated.

Game theory is an obvious model for scenarios with heterogeneous and competitive agents, but searching for the optimal Nash equilibrium is also a useful formulation for cooperative problems. WoLF and related approaches are often applied in such problems, with each agent gradually adjusting to the others so that the whole system is incrementally improved. Although there is no guarantee that the optimal equilibrium is found, the technique is effective, and in large problems it is often sufficient to come up with a "good" solution rather than the optimal solution.

Example 7. Game theory in Tameugny

Twin boys are standing near the old people's home when the rescue workers leave it with all the survivors. During the rescue several parts of the building have collapsed, and more may go at any moment. The boys know that they can dart into the building to loot it for jewellery, but if they do, that will be the signal to the rest of the gang to join them. If the rest of gang are not put off by the risk of building collapse, they will surely all want to join in and claim some of the spoils—greatly increasing the risk of building collapse for all. The twins must decide how likely the others are to follow them in their dash into the building before deciding to make it. A possible set of outcomes for the game is summarised below. In fact, each outcome is associated with some probability and the twins will make their decision by considering all probabilities.

Outcome for the twins:

		Twins	
		Loot	Don't Loot
Gang	Loot	BC; -100	S; -20
	Don't Loot	J; 100	0

Outcome for the gang:

		Twins	
		Loot	Don't Loot
Gang	Loot	BC; -100	J; 15
	Don't Loot	0	0

Outcome *Building collapse* $BC = -100$

Outcome *Jewellery* $10 < J < 100$

Outcome *Loss of status* $S = -20$

2.3.3 Integrated Learning and Coordination

Considering these coordination techniques in the light of our domain requirements, we believe that "acting" and "coordinating" in uncertain systems should be completely integrated. That is, rather than use an explicit coordination layer, agents should include their beliefs about other agents' behaviour in their action selection mechanism, and adjust their own action according to their beliefs about the other agents. By doing this, agents can smoothly make decisions about coordinated actions. Moreover, we believe that such an integrated approach should be based on sound theoretical principles, allowing us to reason about the behaviour of agents. In uncertain and dynamic domains, this motivates the use of multi-agent learning models, since these provide a basis for such coordinated action selection and are designed for uncertain domains. Furthermore, although in large domains it may be impractical to learn a complete solution in real time, we have explained that learning methods can be used on top of other coordination mechanisms to provide adaptability on top of known conventions or communication languages, to select between coordination mechanisms, or to use learning for some subproblem. We therefore explore the application of multi-agent learning models to dynamic, partially-observable domains. Finally, we observe that within model based learning it is sensible for agents to maintain models of the other agents separately from the environment, as these models need not be treated as Markovian. Therefore, the game theoretic paradigm, computing "best responses" to agents within their environment, is appropriate and more flexible than treating other agents implicitly.

2.3.4 Learning Models of Other Agents' Behaviour

Given the aim of learning models of the environment, we have previously discussed reinforcement learning. However, learning about other agents' behaviour is typically a different kind of task from learning about the environment. In a fully observable domain with the Markov assumption, the optimal action will only ever depend on the current state. Therefore, agents can learn simple models of the strategies of the other agents, using multinomial distributions over actions (one for each state) and updating these distributions either using a simple frequency count or using Bayes' rule. In the situation in Example 7, this may mean the twins observing the state *(Home empty, Gang in alley)* and deciding to loot, or observing the state *(Home empty, Gang at head of alley)* and deciding not to risk it. This is known as *fictitious play* [17]. Conversely, in scenarios where the full state is unknown to the agent, simple fictitious play is not appropriate. Each agent may have knowledge of the environment and a model of the current world state—but this is not sufficient to respond optimally to the other agents. In a rescue scenario, some rescue tasks require several agents, and so the agents must come to the same conclusions about when these tasks are approached. If agents have differing views of the situation, they may not make the same decisions about urgency, resulting in an ineffective dispersal of agents. In Example 7, the twins may believe mistakenly that the gang will realise how unstable the building is, and thus expect the gang to take more care than it does, or they may not know how desperate one of the gang is for cash.

In principle, each agent can maintain and update a POMDP in which the unknown POMDP "state" includes the world state, the other agents' world models, and behavioural models for the other agents. In practice, it is not tractable either to update such a model or to determine a best response within it without performing some approximations—for example, projecting just a small number of steps into the future, and using a domain-specific heuristic to estimate the values of those future states [14]. Even this heuristic approach relies on each agent being able to predict the computations of the other agents—each must be initialised with the same random seed. A different approach to approximation is to restrict the possible opponent strategies to those which can be described by regular automata, often called *finite state machines* or *finite automata*.

2.3.5 Finite State Machines

An agent controlled by a finite state machine has a number of internal states, each associated with an action (or a probability distribution over actions)—this tells the agent how to act when it reaches this internal state. After taking an action, the agent's observations determine its movement to a new internal state. The finite state machine captures the notion that an agent's beliefs can be approximated, for the purposes of decision making, by a variable but finite sequence of past observations, and examples such as [38] [6] demonstrate that it can be very effective. Furthermore, approximate best responses to finite state machines can be computed efficiently [23].

To date previous work using finite state machines focuses on offline solutions to multi-agent problems, precomputing responses to every possible belief state. However, it is impossible for every belief state to be reached: each belief state which is visited narrows the space of possible future beliefs (at least within a static environment). For offline solvers without tight time constraints, there may be no problem in generating redundant information. Other approaches use the intuition that the belief space need only be divided into sufficient chunks to determine the next action, for example using principal components analysis on a discretized state space [31]. The alternative to such techniques is to search for solutions online. This is the only way of approaching very dynamic systems, or systems where the problem parameters may not be known in time to perform a comprehensive offline search—as is likely to be the case in our target domain. Online solutions will, of necessity, be approximate, since any accurate solution projects infinitely far into the future and thus is effectively an offline solution.

In the next section we expand some of these ideas and describe in detail an algorithm for online cooperative action in partially observable multi-agent systems in which agent communication is limited to information-sharing. Our algorithm uses finite state machines to model the policies of the other agents and each agent computes online a best response to its beliefs about these finite state machines.

3 Bayesian Learning Models

As outlined in the previous section, we will use finite state machines to model individual agent policies in a multi-agent setting. In this section we flesh out the theoretical background behind this model, first outlining an exact theory and then showing how this is approximated by a finite state machine model. Then in Section 4 we describe the finite state machine model in more detail.

3.1 Bayesian Learning

We begin by specifying our definitions. Throughout, we assume that there is some underlying world state, s, which changes in response to the joint actions of the agents. The progression of world states and joint actions forms an MDP. We assume that agents are not able to perceive s completely, but make some observations o from which they make inferences about the state. These observations may include communications from other agents—we do not treat those distinctly in this work. More formally, we will make use of the following definitions:

- $S : \{s_0, \dots s_{n_s}\}$, a set of states. A state will generally be described by a set of state variables.
- $I : \{I_1, \dots I_k\}$, a set of k agents
- $L \subset S = \{L_1, \dots L_k\}$, a location variable for each agent. These determine the viewpoint from which agents make local observations.
- $A = \{a_1, \dots a_{n_a}\}$, the set of individual actions. $\mathbf{A} = A^k$ is the set of joint actions. Thus, we differentiate between a single action a and a joint action $\mathbf{a}$ by using bold for the latter, to emphasize that it is a vector. We may also use $\mathbf{a_{-i}}$ to refer to the vector $\mathbf{a}$ with the element corresponding to agent i removed, and $\mathbf{a} \circ a'$ to refer to $\mathbf{a}$ with a_i integrated.
- $O : \{o_o, \dots o_{n_o}\}$, a set of observations
- $T_f : T_f(s_{t+1}, s, \mathbf{a}_t) = P(s_{t+1} | s_t, \mathbf{a}_t)$, the transition function from state to state, where $s_{t+1}, s_t \in S$ and $\mathbf{a} \in \mathbf{A}$.
- O_f: An n_o-dimensional function where $O_f(s_t, o_t)_i = P(o_t | i, s_t)$, the observation function for agent i, where $o_t \in O$ and $s_t \in S$.
- $R : \{r_1, \dots. r_{n_r}\}$, $n_r \leq n_s$, a set of possible rewards which an agent may receive

- $R_f : SxAxS \rightarrow R$, a reward function, specific to the agent. Typically, the reward will be associated with the immediate state, but for some problems it may be associated with the transition between states (for example, if actions have a cost).

When taken together, T_f, R_f and O_f describe the dynamics of the environment. We may use $\theta = (T_f, R_f, O_f)$ to refer to these dynamics as a whole. An individual agent, A, may also have:

- A (deterministic) policy $\pi : (p, h, o_t) \rightarrow a$ where p defines any prior or domain knowledge, h defines all relevant historical information (observation sequences including communications from other agents), $o_t \in O$ is the current observation and $a \in A$ is a single agent action. Typically, (p, h) will be compressed to contain the sufficient statistics for a belief state (a probability distribution over states and unknown parameters).
- Beliefs over unknown parameters: for some variable X taking values $x_1, x_2, \ldots$, $b(x_i)$ is the probability that $X = x_i$, given the agent's prior information and subsequent observations.
- Models of the other agents' behaviour: $P(\pi_i | p, h)$ where π_i has the same form as π above and (p, h) refer to the prior and historical information of the agent A. To be clear, we assume that the other agents have deterministic policies, and our agent maintains *beliefs* over these deterministic policies.

Taking these definitions, we go on in the rest of this section to build up a formal model of learning in multi-agent systems. The following section (Section 4) explains an approximation which can be used to make implementing this model practical in a particular special case of interest to us. First, however, we introduce *Markov Decision Processes*.

3.2 MDPs and POMDPs

The transition function T_f has the *Markov property*: the probability of future states depends only on the current state and the action choices, and not on past state history. Consequently, $\{S, A, T, R\}$ defines a *Markov Decision Process* (Figure 2):

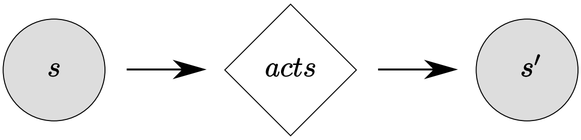


Fig. 2 Markov Decision Process progression

In choosing an action at time $t = t_T$, the agent's aim is to optimise the expected discounted future rewards, defined by:

$$R_T^\gamma = \sum_t \gamma^t r_t \quad (t \text{ ranges from } T \text{ to } \infty) \tag{1}$$

where $r_t \in R$ is the reward at time t. γ is a problem specific parameter which defines the agent's *myopia*; that is, to what extent it considers delayed future rewards to be important. It balances the importance we place on future states with our need to accumulate reward now. In practical terms it will be chosen to express the extent of lookahead appropriate to the problem (consider chess as an analogy: for the most part, say, 3 steps of lookahead are sufficient to play well). Typically, we will use a γ value of around 0.8, making lookahead negligible after around ten steps into the future—in a fragile disaster scenario we expect this to be sufficient for most planning purposes. It is most common for reinforcement learning algorithms to set γ between 0.7 and 1, although the choice will depend on the exact problem.

Now, in a fully observable world, $O = S$ and $P(o_t|s_t) = (1 \text{ if } o_t = s_t, 0 \text{ otherwise})$, i.e. the agent knows the complete state s_t at every timestep t. Given the *Markov property*, its optimal policy therefore need depend only on the current state. We can therefore define a policy in a fully observable MDP by $\pi(s) = a$, a function from states to actions. Then, if the strategies of the other agents are known, the agent can compute its own optimal policy by solving the large simultaneous equation known as the Bellman Equations (2 and 3), via *dynamic programming*, and then taking the policy π^* described in equation 4.

In more detail, $Q_\pi(s,a)$ is the (discounted expected) value of taking action a from state s, and then working to policy π. $Q^*(s,a)$ is the (discounted expected) value of taking action a from state s, and then working to the optimal policy π^*. We will use "best response" to refer to the optimal single-agent action, a, maximising $Q(s,a)$ throughout this paper as we replace s with more complex models.

$$Q(s,a) = \sum_{s'} P(s'|s,a)[r(s') + \gamma V(s')] \tag{2}$$

$$V(s) = \max_a Q(s,a) \tag{3}$$

$$\pi^*(s) = a \text{ such that } Q(s,a) = \max_a Q(s,a) \tag{4}$$

$$P(s'|s,a) = T_f(s', s, a \circ \mathbf{a_{-i}}) \tag{5}$$

where $\mathbf{a_{-i}}$ is the joint actions of the other agents as defined by their strategies

There are various ways of efficiently approximating these solutions in large problems, and for solving in continuous systems. Briefly, the equations can be solved iteratively, and efficiency is achieved by (a) updating the states most likely to have changed first, and (b) updating "nearby" states when a state is updated [36]. We do not go into details of these solution techniques as realistically we are unlikely to know all the necessary parameters. In the next section we explain how this model is extended into systems with unknowns.

3.2.1 Partially Observable Systems

It is often the case that the agent may not know (in the case of static parameters), or be able to observe (in the case of state-related values) all the details of the MDP. If the underlying state s cannot be observed, then the problem becomes a *POMDP*: a "partially observable" Markov decision process (Figure 3). At each step, the MDP proceeds behind the scenes, while the agent makes observations o derived from the underlying state s, where o is insufficient for the agent to reliably determine s. $O_f(s,i)$ describes the probability density function $P(o|s)$ for agent i.

To solve this POMDP, we can derive from it a secondary MDP—a *belief MDP*. The multi-dimensional states of this secondary MDP have one continuous variable,

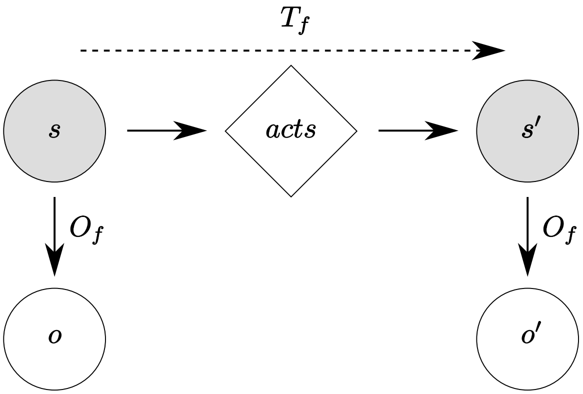


Fig. 3 Partially observable Markov Decision Process

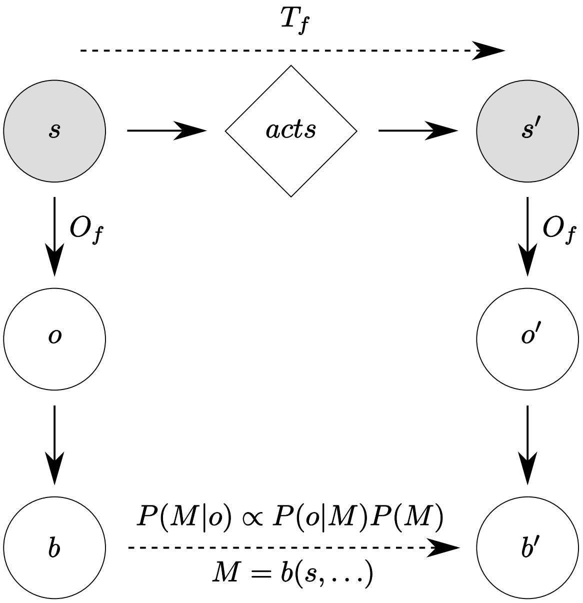


Fig. 4 POMDP inducing a Bayesian belief state MDP

$b(s)$, for every possible value s of the underlying state. The value of $b(s)$ indicates the probability that the underlying state is s, given the agent's prior knowledge and the history of observations and actions. The system proceeds from b to b' at each step using Bayes' rule (equation 6) to update the state probabilities (Figure 4):

$$P(M|observations) \propto P(observations|M)P(M) \tag{6}$$

This belief MDP is, therefore, completely known, and although continuous and high-dimensional has an exact solution describing the optimal action in any belief state. This solution will inherently take into account the need for exploratory actions.

In principle, any general techniques for continuous MDP solutions can be used to solve the belief MDP [36]. However, all belief-state MDPs fall into a particular class of continuous MDPs, since each belief state restricts the possible future belief states. More efficient solution techniques exploit the properties of these MDPs [28] [27].

Given this, we can extend the belief MDP idea further to consider cases where the environmental dynamics, θ, are not known or are partially known. In these cases, we can consider an underlying MDP which has the dynamics, θ, as one of its state variables. This MDP has a known transition function: $(s,\theta) \rightarrow (\theta(s),\theta)$. The observations for the POMDP associated with this MDP will include state transitions as well as the immediate observations. In principle, this POMDP can be solved exactly as described above. Finally, the same model extends into the multi-agent world by including the actions of other agents in the underlying state, and the behaviour functions of other agents in θ. In a partially observable system, the behaviour of another agent will depend on its beliefs about the state, and so we also add the beliefs over states of the other agents to our own MDP state:

$$s_{MDP} = \{s, \forall j.(\sigma_j), \forall j.(b_j(s)), \theta\}$$

To date, existing work has studied some sub-cases of this general model: the fully-observable case where the dynamics are unknown, for single agent problems [11] and multi-agent problems [8]; and the partially-observable case where the dynamics are unknown for multi-agent problems [30]. All of these find online solutions using appropriate approximation techniques. In particular, in solving the Bellman equations, typically these techniques will only refer to a small number of belief states, beginning at the current one. Recall,

$$Q(s,a) = \sum_{s'} P(s'|s,a)[r(s') + \gamma V(s')]$$

The belief-state version, writing b for the belief state and leaving s to refer to the underlying state, is

$$Q(b,a) = \sum_{s'} P(s'|b,a)[E[r(s')|b] + \gamma V(b')]$$

where b' is the belief state resulting from the transition to state s'.

Finally, the case of particular interest to us, the partially-observable multi-agent case with known dynamics (sometimes described as a partially observable stochastic game, or POSG) has also been investigated. For example, in one online approximate algorithm [14], each agent tries to compute the joint optimal action for that step, then executing its own part of this joint optimal action. Providing that all agents are initialised with the same information (in particular, they should share a random seed), every agent can compute the approximately optimal action so that the actions are truly cooperative. Although this algorithm is theoretically sound, it is computationally intensive and has only been tested on relatively small POSGs. More recent work has investigated offline algorithms for a special case of much larger POSGs, the networked POSG. In the case where the agents are networked according to a specific structure—such as a sensor network—it is possible to exploit this structure to develop more sophisticated strategies for agents located in critical parts of the network, and simpler strategies for agents located in less critical regions [23].

An alternative technique for making approximate action choices is to gather together similar states, belief states or groups of observations, reducing the state space.

In particular, in problems where a notion of proximity can be defined between states, an action can be decided for a new state based on experience of nearby states. Examples of the former include manual feature abstraction and hierarchies [15]. Examples of the latter include include neural networks [36], Kohonen maps [35] and belief compression via principal components analysis [31]. We leave investigation of such state aggregration to future work.

A further optimisation that has been used to good effect for modelling and responding to agents in partially observable domains is to restrict agent policies to the class of policies which can be described by *finite state machines* (sometimes called *finite state automata* or *regular automata*, and abbreviated to FSMs). Using this class of policies, which we describe in more detail in the next section, it has been shown that it is possible to compactly represent good approximates to the optimal agent policy [6] [9]. More recent work has used finite state machines to find offline solutions in partially observable domains [23]. In the next section we describe finite state machines in more detail and develop an online solution strategy to partially observable problems, using finite state machines to model the behaviour of the other agents.

4 Bayesian Learning Approximation Using Finite State Machines

A finite state machine can be used to represent an agent's policy. We have discussed fully-observable MDPs in which the agent's policy is decided on its immediate observations, and partially-observable MDPs in which the agent state is a continuous, high-dimensional belief-state derived from its entire history. A finite state machine policy falls between these two: the agent state is based on a variable length history. A fixed and finite number of agent states, more than the number of possible observations, are defined in the finite state machine and the agent moves from state to state of the machine based on its observation.

We now detail how to model agent policies using finite state automata (4.1 and 4.2). In Section 4.3 we then explain how these models fit with the multi-agent POMDP solution techniques described above, giving an algorithm for online learning (4.4) and explaining how this model extends previous work in the area. First we begin with the definition of a finite state machine.

4.1 Definitions

A deterministic finite state machine has:

- A set of n nodes $N = \{n_1, \ldots n_n\}$
- A set of m edges $E = \{e_1, \ldots e_n\}$
- For each node, an associated action a from the set of actions
- For each edge, an associated observation o from the set of observations

One of the nodes is designated as a start node, N_0. We write $Act(n)$ to refer to the action associated with a node n.

An agent's policy is determined by such a state machine (Algorithm 1.): at each node (or agent state), the agent carries out the associated action. The resulting observations determine the agent's transition to a new node within the FSM.

Algorithm 1.. A finite state machine policy

1. The agent begins at the start node n_0.
2. The agent performs the action associated with the current node n.
3. When all agents have performed their action, the system moves to a new world state s, supplying agents with observations o.
4. The agent moves along the edge associated with o, arriving at a new node n'.
5. Repeat from step 2.

Now, in order to use finite state machines as representations of agent policies in unknown multi-agent scenarios, we will do two things: (1) to learn the finite state machine models over time, from the sequence of observed actions and state observations, and (2) to derive an online policy as a best response to a set of (beliefs over) FSM policies. We describe each of these in turn, bringing them together in Section 4.4.

4.2 Learning FSMs

In principle, learning a deterministic finite state machine from a set of observations can proceed as follows [6]:

- **Base case:** initialise the FSM with the single node N_0, setting the associated action to the first observed action
- **Recursion step:** given a FSM and an observation string, determine if the observation string is consistent with the FSM:

  1. Find a node whose action corresponds to the first action in the string: *if there are no untested nodes remaining, FAIL*
  2. Follow the FSM as prescribed by the observation sequence until (a) the action associated with a particular node does not match the action in the sequence: *FAIL, return to 1* or (b) the end of the sequence is reached: *CONSISTENT*

  If the observation string is consistent, then no further action need be taken. If the observation string is inconsistent, then we select a node from one of the failure points, and expand the FSM to include the new string.

Then, given a FSM and a particular (short-term) observation history (after applying the above algorithm to the history), we can construct a list of possible current nodes for the corresponding agent by considering each of the starting points consistent with the observation history and following the FSM through to a current node from each (abandoning any inconsistent nodes *en route*). The probability of each resulting current node will be the total probability of all start nodes which reached it, with that probability having been computed in a previous step.

However, there are two problems: one is that observation strings can be of indefinite length, i.e. we may find ourselves storing the entire observation history in order to accurately build the FSM. The second is that although the FSM is a deterministic model, the behaviour it is modelling may be neither deterministic nor static. (A third issue is that we do not in fact know the observation strings, but rather have probabilities over them which are based on our own observations). We therefore wish to adjust our learning strategy to take these facts into account.

A point to note is that although we do not know the strategies of others or their optimal strategies, because we do know the MDP and the observation function, we can make some judgements about how much observation history is likely to be important in making decisions, providing us with a way of judging the optimal size of the FSMs.

We propose to sample possible observation strings from our belief state, and construct a candidate FSM for each sample, using the following tactics in learning these candidate FSMs:

- Define a maximum number of nodes which can occur in the FSM
- Break the observation history into overlapping observation strings of length l
- Assign each observation string a *likelihood* based on the frequency of occurrence and its sample probability, weighting more recent occurrences more highly. Discard completely observation strings older than n_t timesteps.
- Rather than resolve inconsistencies by always creating new nodes, resolve inconsistencies by appealing to the likelihood of each of the inconsistent strings, and discarding the least likely

In the next sections we describe in more detail an algorithm for learning FSMs from observation strings.

4.2.1 A Polynomial FSM Learning Algorithm

For any set of agent behaviours, there may be several possible FSMs. The least compact FSM for a finite time period has a distinct node for every time step. The *minimal* FSM for an agent's behaviour has the smallest number of nodes necessary to describe the behaviour exactly. Now, finding the minimal FSM is an NP-complete problem and cannot be approximated by any polynomial-time algorithm [6]. However, it is possible to learn compact FSMs in polynomial time, for many practical problems. The US-L* algorithm [6] has polynomial running time and has been shown to be effective at finding compact models of agent behaviour on small agent coordination problems—we propose to test it on larger problems.

This algorithm models the FSM using a table, with rows corresponding to observation string prefixes s, columns corresponding to string suffixes e, and the table entries corresponding to actions σ. The alphabet of possible observations is Σ. The table is then partitioned into equivalence classes:

$$C(s) = row(s) | row(s') = row(s)$$

The table must be constructed in such a way that it describes a FSM: that is, it must be

- **consistent**: $\forall s_1, s_2 \in S, [C(s_1) = C(s_2) \rightarrow \forall t \in \Sigma, C(s_1 t) = C(s_2 t)]$.
- **closed** : $\forall s \in S\Sigma, \exists s' \in S, s \in S, s \in C(s')$

From such a consistent and closed table a deterministic FSM can be described.

Specifically, US-L* marks entries in the table as either *hole* entries or *permanent* entries. The former are those which can be reassigned as the algorithm tries to readjust the table for consistency. Only when no hole entries can be reassigned is a new test added to the table. Permanent entries correspond to a fixed action.

The algorithm proceeds by:

- Take a set of observation strings
- Initialise the table so that all the prefixes of the observation strings have an associated row in the table, and there is just one column with the empty string.
- Fill in the table entries using the observations, marking entries as *hole* entries if they are not supported by previous examples or *permanent* entries if they are are supported by previous examples. In order to bound the size of the automaton, we specify a maximum number of times a *hole* entry can be changed, basing the maximum on domain knowledge if it is available: the maximum should depend on the dynamism in the system (since an entry will change if the system is changing) and on the uncertainty in the system. In our work, we may adjust the maximum over time using learned domain knowledge.
- Adjust the table to make it consistent, adding new columns to the table where necessary (adding a new column enables the separation of one equivalence class into two—this adds at least one new state to the corresponding automaton).
- Adjust the table to close it, adding new rows where necessary.
- Take the next set of observation strings and loop.

This algorithm is designed to be used as an online algorithm for an adaptive agent to learn models of opponent behaviour, although Carmel and Markovitch only apply it to repeated two-player games. We will be investigating its application in our domain, specifying in advance a maximum size for the automata. Now, in order to make use of these finite state machine models of agent behaviour, our agent (maintaining these models) must be able to find an optimal response to what it believes to be the current situation. Referring back to our generic Bayesian model, this means evaluating $Q(b,a)$ for a belief state b which includes beliefs over finite state machines. The next section explains how this is done.

4.3 Online Solutions: Best Response

Previous work [38], [6] has considered fully-observable, but non-Markov, repeated games. In such scenarios finding a best response is straightforward, since the state and consequently the reward can be computed for every step.

By contrast, in our work, the state is not known. This adds to the complexity of the situation (as previously discussed), since even if the policies of the other agents are known, we do not know what observations they may make and consequently cannot determine what their actions are based on. We consider first this idealised case where the policies of the other agents are known. Now, we can compute a best response to any belief-state b using the Bellman equations, as discussed in Section 3:

$$Q_i(b,a) = \sum_{s'} P(s'|b,a)[r(s') + \gamma V(b')] \tag{7}$$

$$\text{where} \quad P(b'(s')|b,a) = \sum_{\mathbf{n_j},s',s} P(b'(s')|s,a \circ Act(\mathbf{n_j}))P(\mathbf{n_j},s|b) \tag{8}$$

$$\text{and} \quad P(b'(n'_j)|b,a) = \sum_{n_j,o',s} P(\mathbf{n'_j}|o')P(o'|s,a \circ Act(\mathbf{n_j}))P(\mathbf{n_j},s|b) \tag{9}$$

$$\text{where} \quad P(o'|s,a \circ Act(\mathbf{n_j})) = \sum_{s'} P(s'|s,a \circ Act(\mathbf{n_j}))P(o'|s') \tag{10}$$

($P(\mathbf{n'_j}|o')$ is 1 or 0.)

In an online algorithm, $Q_i(b,a)$ can be approximated from the current belief state by projecting k steps into the future. On the kth step, we replace $V(b')$ in equation 7 with some heuristic value. Possible heuristics include 0, a Q_{MDP}-based heuristic [19] or some domain-specific heuristic (for example, the expected distance from any agent to a goal, or visible future rewards such as victims which can be saved, in a disaster problem). Algorithm 2. outlines this best response solution. Such *finite horizon* algorithms have been used in related belief-state problems in many cases: in observable problems with unknown parameters (the heuristic is to assume that the current parameters are correct, and solve the corresponding MDP [8]), in finding offline solutions for networked POMDPs [23], and in online partially observable stochastic games [14].

Algorithm 2.. Finite-horizon best response

- At timestep t, agent i has beliefs b over the state and the nodes n_j of the other agents j, and knows policies $n_j \rightarrow a_j$ for these agents.
- For some k, compute $Q_k(b,a)$ for each possible action a, using
  - $Q_k(b,a) = \sum_{s'} P(s'|b,a)[r(s') + \gamma V_{k-1}(b')]$
  - $V_k(b) = \max_a Q_k(b,a)$
  - $V_0(b,a) = \sum_{s'} P(s'|b,a)V_{heuristic}(s')$
- Execute the action a which maximises $Q_k(b,a)$

In our partially observable setting, where the agent does not in fact have knowledge of the policies of the other agents, but rather has beliefs over these policies, we propose to estimate the best response to the belief state by sampling from the possible policies to obtain a selection of sets of FSMs, $F = \{F_1, ...F_m\}$. For each sample FSM set F_s (containing a FSM for each other agent), the agent computes a best response action $BR_i(F_s,b)$. The action decision is then given by:

$$a = max_{a_i} \sum_{i=1}^{m} P_i.\delta(BR_i(F_s,b) = a_i)$$

(where $\delta(A = B) = 1$ if $A = B$ and 0 otherwise).

4.4 An Online Learning Algorithm

We conclude this section with a complete description of our algorithm, which brings together several of the techniques described above. This is an algorithm implemented by a single agent who is aiming to adaptively find a best response to the behaviour of the other agents in the system. Our intent is that when all agents are implementing this algorithm, adapting to each other, they should converge on a "good" collaborative solution for the problem. This algorithm, as described below, maintains approximate models of the other agents in the form of finite state machines. A set of possible models is held in a belief state which is updated using Bayesian learning. At each step, the agent uses its observations to update each of the possible models and to update its beliefs about the world. It then computes the finite-horizon best response to each of these possible models and weights the possible responses with its belief in the corresponding model to decide on its action.

- An agent maintains a current belief state, $b(X)$, with beliefs over the variables $X = (s, \{o, F, n\})$ where s is the current state, and $\{o, F, n\}$ describes a set of triples: in each triple, o is an observation history and (F, n) are the induced FSM and current node in the FSM. The belief state contains one such triple for each other agent in the system. The agent also maintains historical information about $b(s)$ over a fixed number of steps.
- Several parameters are fixed initially: F_{max} the maximum number of nodes in any FSM, γ the myopia of the agent, n_t the horizon length to use in computing an approximate best response, o_l the observation window length. n_t may be determined based on γ: roughly, for a state n steps into the future, s_n contributes $\gamma^n.r(s_n)$ towards the discounted future reward. Thus with $\gamma = 0.8$ (a common myopia value), after 10 steps less than 10% of the reward will be contributing towards the estimates of the future reward. This may be a small enough value to ignore. If γ is increased to 0.9, then it will take 21 steps before the fraction of the reward under consideration is reduced below 10%.
- **initialise:**
  The belief state is initialised: $b(s)$ is initialised either to uniform beliefs or biased based on domain knowledge. The observation strings o are all empty, and the F have a single node with uniform probabilities over all actions[1]
- **at each step:**
  - The agent observes the actions of the others and makes observations about the state: these observations are used to update $b(s)$ using Bayes' rule.
  - The observation samples o are extended into the current time frame to obtain o', reweighting as appropriate. This is achieved by sampling from the expected observations of the other agents, given the current observation samples and $b(s)$. When the length of an observation string exceeds o_l, the earliest observations are dropped. If a sample's likelihood falls below probability threshold p_s, the sample is discarded, and a new string sampled using $b(s)$ and the stored history of $b(s)$ over o_l previous steps.
  - For each observation sample o', update the FSMs F associated with the sample with the new information in o' using US-L*. The weighting given to the FSM F is the probability of the associated observation sample.
  - For each sample FSM, compute an approximate best response, and thus decide the maximum likelihood best response action a from the FSM weightings as described in Section 4.3.

[1] It would be possible to initialise with a more sophisticated set of F corresponding to shared conventions relating to the domain, for example encapsulating the knowledge that agents will run from a burning building. We leave that possibility to future work.

- Perform the action *a*
- Repeat

To reduce computational requirements, rather than doing all of this every step, we may prefer to collect behavioural samples over several steps and update our model less frequently. The best response is still computed every step.

Thus, in this section we have outlined a theoretical model for the online solution of partially observable multi-agent systems, based on the POMDP model, and then shown how we can approximate a particular (challenging) case of this model using finite state machines to model agent behaviour. In order to demonstrate the effectiveness of this model, we have implemented it on a rescue problem. In the next section (5) we outline the problem before going on to describe our results (Section 6) and how they compare with the state of the art.

5 Model Instantiation

In order to test the algorithm on a challenging problem, we implemented a rescue scenario involving coordinating ambulances. We compared our algorithm with a current state of the art algorithm and a hand-written solution for this problem. In this section, we specify the problem as a multi-agent POMDP and explain how we simplify the observation space.

In more detail, in the rescue problem we have an *n* by *m* gridworld. *k* agents can move left, right, up or down (constrained, of course, at the edges of the grid). In the gridworld are buried victims, described by two parameters: *D* and *R*. *D* (‘deadness’) is a measure of the proximity of the victim to death. When it reaches a maximum level the victim is dead and subsequently ignored for the purposes of the rescue problem. *R* (‘rescue needed’) is a measure of the depth at which the victim is believed to be buried. Agents digging can reduce R. If R reaches 0 before the victim dies, then the victim is assumed to be safe. The urgency of the victim therefore increases with increased D and with increased R, unless R is sufficiently large compared with D that the victim can be considered a lost cause. Figure 5 shows one step on the grid for a 4x4 grid with three agents.

Specifically, taking the model of Section 3, the various parameters are instantiated in the following way:

States: A state of this world is described by using a pair of variables for each of the grid squares, characterising the *D* and *R* values in the square (we make the simplifying assumption that there can be at most one victim in the square), and a variable for each agent, identifying its current square. We use l_d and l_r discrete levels to describe *D* and *R*, so for each square there are l_d*l_r possible states, and for each agent there are $m * n$ possible states, making a total of $((l_d * l_r)^{(m*n)})$ possible states.

Agents: We assume that the number of agents, *k*, is fixed throughout each problem and known to each agent.

Locations: The location variable for each agent is its current square.

Actions: Agents may take Move actions (left, right, up or down), or Dig actions in their current square.

Observations: An agent observes some subset of the state variables, so there is one observation variable for each state variable. The values taken on by observation variables are those of the corresponding state variable, plus “null”.

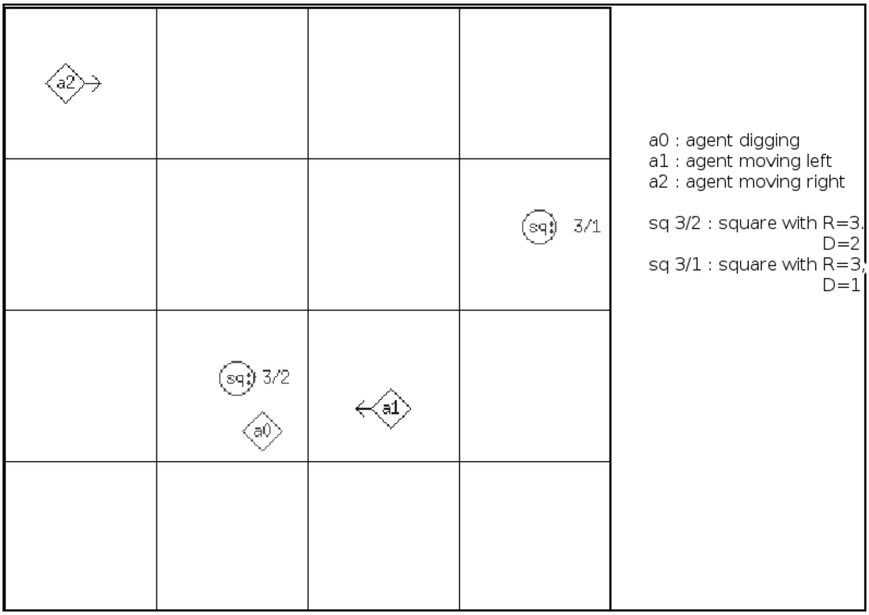


Fig. 5 One step of the rescue problem on a 4x4 grid with three agents

Transition function: Move actions move the agent one square in the requested direction, unless this is impossible in which case the action has no effect. Each square transitions (D,R) independently of other squares, so it is sufficient to define the transition function for one square. We use two global probabilities, p_d and p_r, to specify the probability of the D level changing (this is a constant probability independent of the action) and of the R level changing if there is a Dig action. If there is no dig action, R remains unchanged. We assume that if there are k digs in a square, they are concatenated. Finally, if a square is empty, we use a further parameter, p_a, to define the probability that a victim will appear in that square. If a victim does appear, the (D,R) levels it has are determined with uniform probability (greater than 0).

Observation function: Agents are able to see the squares (deadness, rescue-level, and any other agents in the square) to the left and right, and above and below them, as well as their own square. Additionally, we define a problem-specific parameter, v, for the visibility. For every other square, the agent will be able to see the agent-deadness D in that square with probability v and the rescue-level R in the square with independent probability v. Since all agent actions are fully observable, we assume that we can also observe all agent locations. This 'visibility' parameter could be justified as some level of communication with a centralised observer, say a helicopter viewing the scene. We assume no error in the observation: either a variable is completely and correctly observed or it is not observed at all.

Reward function: The reward function is a function of both the previous state and the current state. For each square, if a victim disappears because they have died, then the reward is decremented by one point. If a victim disappears because they have been saved, then there is no change to the reward. Consequently, for this problem rewards will always be less than or equal to 0.

The above definitions allow us to define beliefs over the values (D,R) of a square (and thus over the state, since locations are observable), and beliefs over the observations of other agents, given their locations:

Agent locations: We are certain for all squares how many rescue agents they contain / for all agents where they are located

The square is observed: We are certain of both its parameters

The square is not observed and has not been observed for i timesteps: for each property sq which may take values x,

$$P(sq_t = x_t | sq_{t-i} = x_{t-i}) = \sum_x P(sq_t = x_t | sq_{t-1} = x) P(sq_{t-1} = x | sq_{t-i} = x_{t-i})$$

where the 1-timestep probabilities depend on p_d, p_r, p_a as appropriate, and the dig observations in that square.

The square has never been observed: This is just as above, but with $P(sq_0 = x_0)$ set to the problem-specific prior probabilities. Here, we assume that all squares are empty to begin with.

The above equations describe our beliefs about the world state: that is, the D and R values of the squares and the locations of the other agents. Similarly, we must define our beliefs about the observations of the other agents. Just as our beliefs about the state of each square are multinomial, the other agents' beliefs about the state of the square will be multinomial. Therefore, in the full POMDP model, our beliefs about other agents' beliefs over the state of the square would take on corresponding Dirichlet distributions. However, we are not trying to maintain beliefs about the other agents' belief states, only about their observations. Now, our own beliefs about the state of the square define exactly what we believe other agents will see if they see that square, as the observation function is deterministic and consistent for all agents. Because we know the location of the agent, we know of the (up to) four surrounding squares it definitely sees. Finally, we know that there is a v probability it will see any other square. Using this model, we investigate the behaviour of our algorithm on the rescue problem.

6 Experimental Evaluation

In order to test our strategy, we compare it against two other online algorithms: the state of the art for online partially observable stochastic games is the Bayesian game approximation using the finite-horizon approximation technique [14], described in Section 3 ("POSG"). However, for large dynamic problems, this algorithm, which is exponential in the number of agents, proves to be very inefficient and we find that for all but the smallest variants of the rescue problem, POSG is too slow to be useful. Previous work on large dynamic rescue problems of a similar form [26] compares with a handwritten strategy ("smart") tailored to the problem, and we do the same thing. Our handwritten strategy is the strategy that was used by the AladdinRescue team for ambulance distribution in the Robocup Rescue competition, which inspired this problem. The algorithm uses a greedy strategy to allocate ambulances to victims and is optimal in scenarios where (1) no new victims are arriving and (2) visibility is perfect [29]. It is therefore not an optimal strategy for the problem as we have stated it, but is a good approximation, thus providing a good target for our algorithm to meet.

Comparing against these two algorithms, and using the null policy in which agents move randomly, but never dig and so never effect any rescues (“`null`”) as a baseline, we investigate our algorithm, “`best response`” over different parameter settings on the rescue problem, and then focus on the scaling properties of the algorithm. Next, we identify the fixed parameters and then go on to our results.

6.1 Experimental Setup

Following experimentation, we fix the following parameters: $l_d = l_r = 4$, $p_d = 0.15$, $p_r = 0.4$, $p_a = 0.05$, $v = 0.5$. In particular, we felt that the choice of four health and burial levels was sufficient to make the problem interesting without making the state space too huge. The other parameters were selected to generate scenarios requiring cooperation: victims were not arriving so fast that simply digging out the nearest was appropriate, victims might require more than one agent for rescue, and victims could survive long enough to be reached by agents some distance away.

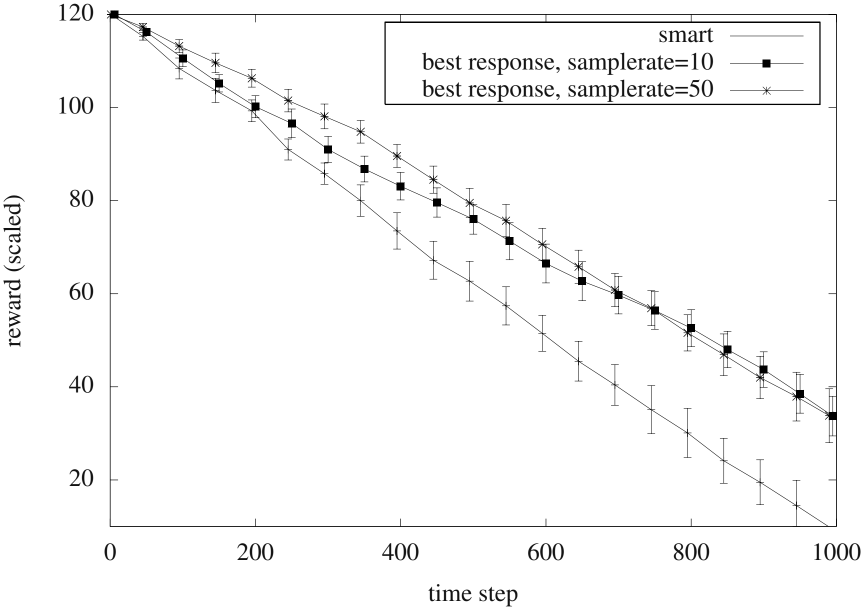
We vary m,n and k as specified. We also experimented with increasing p_a towards problems where “dig nearby” becomes a reasonable strategy, and varying v. Finally, in the belief-state based algorithms, we must take samples from the belief state. We define the *sampling rate* as the number of samples taken for each variable, initialising it at a rate of 35 (for comparison, previous work on a single agent problem found that 20 samples was sufficient for good solutions [11]).

In every experiment, we carried out several runs of the problem, varying the initial placement of civilians and randomising their arrival and visibility. The same random seed was used to initialise each of the test algorithms in each run. The error bars included in the results show the 95% confidence intervals around each point. The rest of this section discusses our key results.

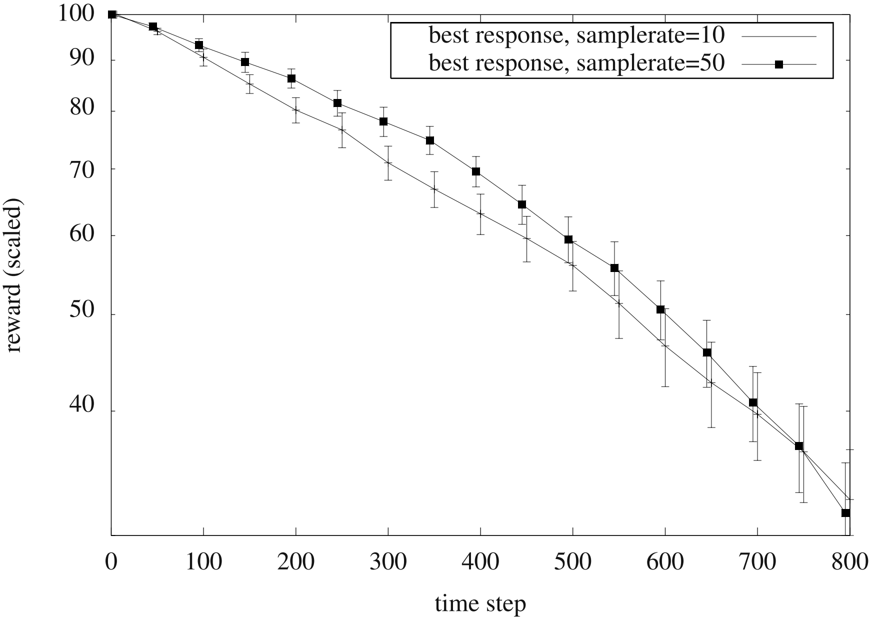
6.2 Examining the Learning Rate

To begin with, we compared the algorithms over the course of 1000 steps on a 7x7 grid, with three agents. We found that the `POSG` algorithm, which is exponential in the number of agents, did not complete in any reasonable time (we consider one minute per step to be “reasonable”for this problem), taking ten minutes for one agent to complete a single step. Figure 6 shows the performance of the `smart` policy with our algorithm over 1000 steps. Our aim was to examine the performance of the `best response` algorithm on a challenging problem, focusing on any changes in its behaviour over time. To this end, we have used two different sampling rates for the `best response` policy, comparing how the agent learns when sampling very little information ($samplerate = 10$) or more information ($samplerate = 50$). We expect that the agent will both perform better, and learn faster at the higher sampling rate.

It is immediately clear from Figure 6 that the `best response` algorithm is outperforming the `smart` policy for these parameters. Now, if our algorithm (`best response`) is benefitting from learning, we expect to see that the advantage the best response algorithm has over the `smart` (handwritten) policy is increasing



(a) Algorithm performing over 1000 steps at two different rates



(b) Closeup of the first 800 steps

Fig. 6 Comparison of two algorithms over time on a 7x7 grid with 3 agents. Note that we use a log scale to show more clearly the differences between the algorithms, and the rewards are scaled up to > 0 for the log scale.

over time. From Figure 6(a) it is not clear that there is a large improvement in this advantage—that is, the lines are fairly straight. However, Figure 6(b) shows a closeup comparison of the two different sampling rates, showing the way in which the lower sampling rate is able to match the performance of the higher sampling rate after around 800 steps. We therefore see that with better information, the best response algorithm is able to perform well on this problem even without accurate models of the other agents, but when the sampling rate is very low, the best response algorithm is able to compensate for this by learning.

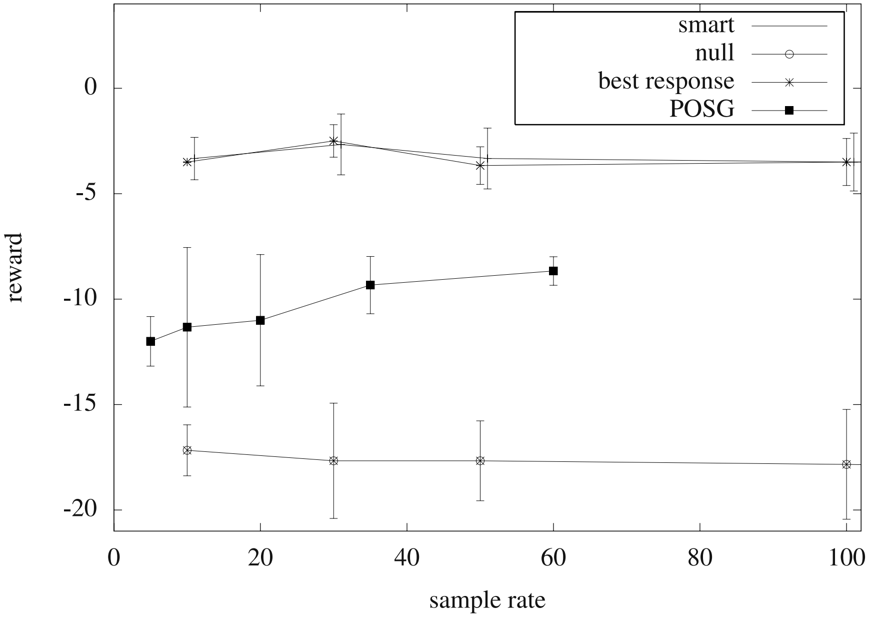
Consequently, it seems that the `best response` algorithm is performing well primarily on the basis of the sampled best response, rather than accurate estimates of the behaviour of the others being critical. In order to investigate further, we compare the algorithms on some smaller problems which the `POSG` algorithm is able to run on, first looking at the effects of changing sample rates in more detail, and then varying two parameters relating to the character of the problem (visibility and victim distributions). This allows us to gain insights into the performance of our algorithm as the problem nature is changed. We also investigate parameters relating to the scale of the problem (number of agents, and size of grid). For each of these experiments we compare the total reward after 150 steps—from Figure 6 we can see that this is sufficient to show the differences between the algorithms or settings.

6.3 Varying the Sampling Rate

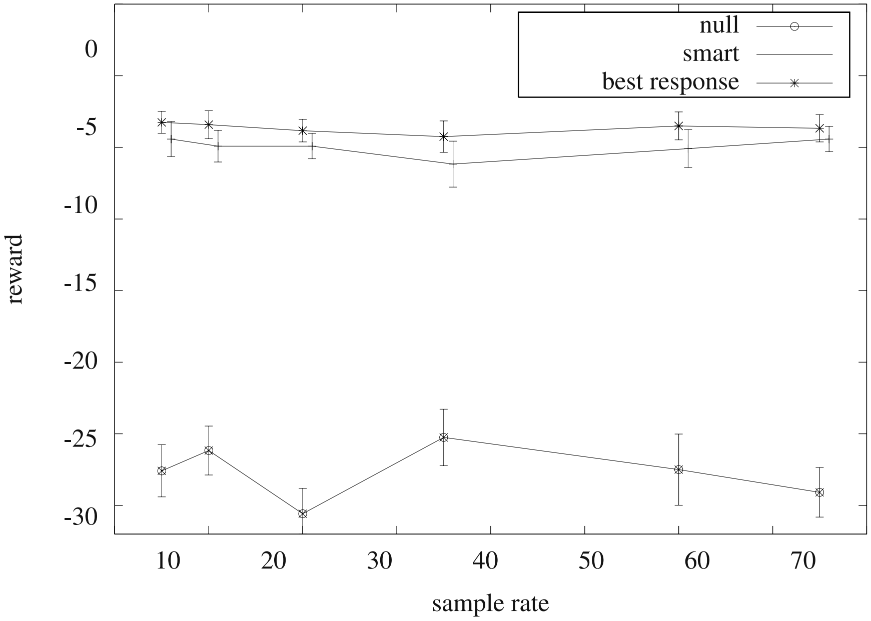
In order to examine how the `best response` algorithm will perform on challenging problems such as those we identified in our domain requirements, we will consider the effects of scale both on solution quality and on the computational requirements. Linked to the solution scales is the number of samples taken in estimating beliefs. The sensitivity of the solution to the number of samples is therefore relevant in considering the effectiveness of the algorithm.

For the `POSG` algorithm, on a 3x3 grid with two agents, Table 8(a) shows the time/sample-rate ratios for 100 steps (to the nearest minute). Since our cut-off was one step per minute, we did not run any tests on the `POSG` algorithm beyond a sample rate of 75, while the `null` policy and the `smart` policy do not do any sampling. For our own policy, which does not need to iterate over all *joint* policies, the scaling factor was much better: Table 8(b) shows the equivalent rates. The `POSG` algorithm is exponential in the number of agents, since it iterates over all joint actions. It therefore scales badly as the number of agents is increased. By contrast, Table 8(c) shows the times for the `best response` algorithm running on the larger problem of a 7x7 grid with three agents. Even on this larger grid the times are well within our “reasonable” range. We next investigate whether there is truly a need for higher sampling rates, since our earlier investigations indicated that the `best response` algorithm is able to perform quite well even at low sample rates.

To this end, Figure 7 shows the effect of changing the sample rate. As expected, neither the `null` policy nor the `smart` policy are susceptible to changing sample rates. However, the performance of the `best response` policy also does not vary much with the changing sample rates. It is also worth remarking that the error does not reduce noticeably as the number of samples is increased, suggesting that the same actions are selected with as few as ten samples. By contrast, the `POSG` algorithm performs noticeably better as the number of samples is increased, and the error around the points reduces.



(a) 2 agents on 3x3 grid



(b) 3 agents on 5x5 grid

Fig. 7 Effects of changing the sampling rate with two and three agents

3x3 grid, 2 agents	
Sample rate	Time
10	12 minutes
20	19 minutes
35	38 minutes
60	67 minutes
75	113 minutes

(a) POSG algorithm

3x3 grid, 2 agents	
Sample rate	Time
10	7 seconds
30	18 seconds
50	27 seconds
100	50 seconds
500	4 minutes

(b) best response algorithm

7x7 grid, 3 agents	
Sample rate	Time
10	18 seconds
35	48 seconds
60	5 minutes

(c) best response algorithm

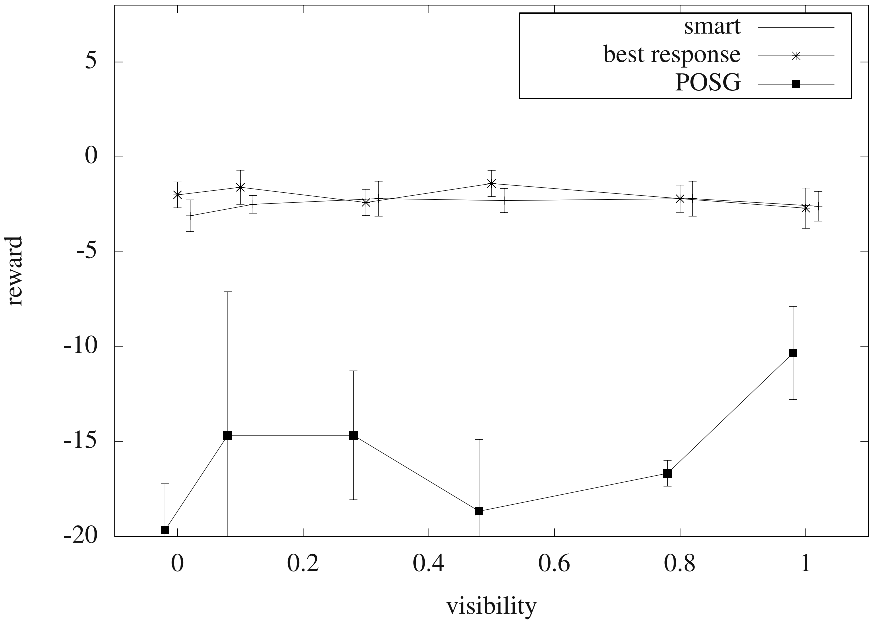
Fig. 8 Time taken to complete one run of 150 steps

These results indicate that similar actions are selected even with a small number of samples, perhaps because the best response can be estimated well, and the best response performs well with small sampling rates, making it possible for the algorithm to be very efficient. This compares favourably with the POSG algorithm which approaches optimality at high sampling rates but performs very badly at low sampling rates, at least for this type of problem. We do not investigate the POSG algorithm in the larger version of the problem (Figure 7(b)) but we see that as for the larger problems above, the best response algorithm slightly outperforms the smart policy, due to its better handling of imperfect visibility. The next section investigates the effects of visibility in more detail.

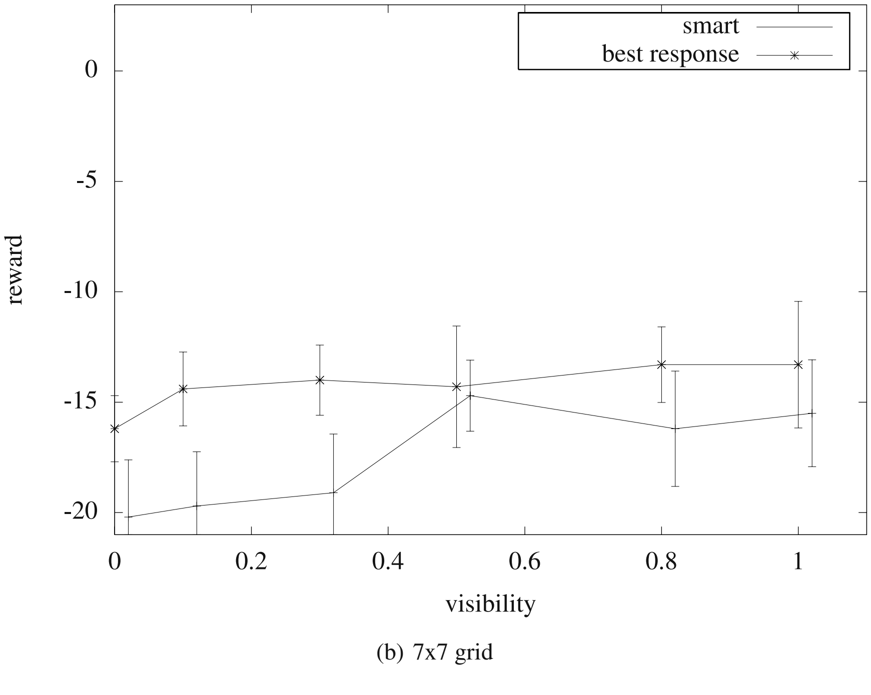
6.4 Varying the Visibility

As the visibility increases and all agents have a better view of the scenario, we expect that the performance of all algorithms will improve. However, we expect the probabilistic algorithms (POSG and best response) to be at less of a disadvantage than the handwritten policy for the lower visibilities—this is because the handwritten policy always behaves as though the visibility is 100% thus does not do any exploration actions.

Figure 9 demonstrates the effects of varying visibility on a 3x3 grid and on a 7x7 grid, each with three agents. In Figure 9(a) we see the performance of the POSG algorithm is much worse than either the smart policy or the best response policy and fluctuates at lower visibilities, but noticeably improving as the visibility is increased. However, both the smart policy and the best response policy do reasonably well even at the lower visibilities, but there is no discernible difference between them. This is because three agents on a three-by-three grid can do fairly well using the very simple strategy of digging where they see victims and can probably directly observe most of the grid between them. By contrast, Figure 9(b) shows the performance on the larger grid. We do not show the slow POSG algorithm on this problem; the baseline of the null policy is at around -90. Here, we see that as expected the best response policy does outperform the smart policy at lower visibility levels, with the smart policy approaching the performance of the best response policy as the visibility increases, although the best response policy continues to outperform the smart policy.



(a) 3x3 grid



(b) 7x7 grid

Fig. 9 Effects of varying visibility

6.5 Varying the Victim Arrival Rate

As well as varying the visibility, we can vary the problem by adjusting the victim density. We expect that increasing the rate at which victims arrive, p_a, and thus the overall density of victims, will make the problem easier, as agents can do well with the simple strategy of digging out the victims around them. The reward for the `null` policy drops sharply—this is because there are more victims dying. This is demonstrated in Figure 10.

As the victim density increases, the optimal strategy approaches the very simple strategy of digging if there are any nearby victims. The point at which the simple strategy becomes optimal is indicated by the point where the `smart` policy stops making improvements over the `null` policy: between the 0.1 and 0.5 arrival rate on the small problem (Figure 10(a)). The `best response` policy has matched the `smart` policy, and the `POSG` policy also catches up by the 0.5 data point. On the larger problem (Figure 10(b)), the `smart` policy and `best response` policy continue to improve across the graph, indicating that there is some sophistication needed in the strategies even at the high victim densities. As expected, on the larger problem, the `best response` policy slightly outperforms the handwritten strategy due to its better handling of the imperfect visibility.

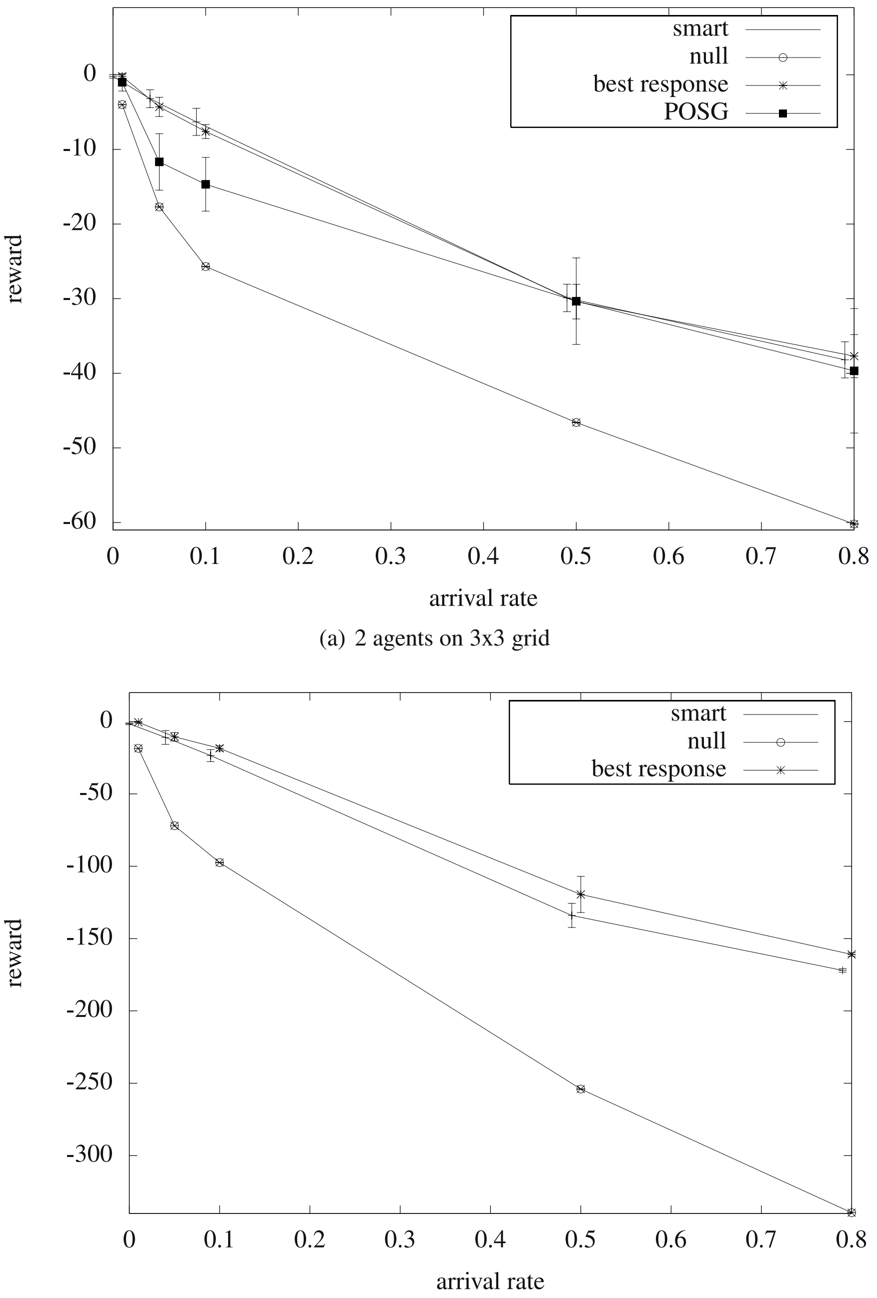
For the next sections, we fix the visibility at 0.5 and the arrival rate at 0.05, as discussed in Section 6.1. We go on to investigate the scaling properties of the algorithms.

6.6 Varying Scaling Factors

The difficulty of the rescue problem scales exponentially with the size of the grid and the number of agents, which are related to the number of states and the number of joint actions respectively. Furthermore, in our implementation, all the agents were running on the same machine as one another and the environment; consequently, the memory requirements of the implementation scaled linearly with the number of agents. Nonetheless, we were able to test our algorithm on grids of up to 12x12 (2^{173} states), and with up to 7 agents (80,000 joint actions).

Now, although 7 agents is not a huge number for an algorithm which we would like to scale into dozens of agents, the primary limiting factor was the memory requirement for our implementation. Figure 11 shows the effect of increasing the number of agents on two larger grids, a 7x7 grid and a 9x9 grid. We observe that on the 7x7 grid as the number of agents is increased, the `smart` policy appears to saturate while the `best response` policy continues to improve. The results are similar for both the 7x7 and the 9x9 grid, although the `smart` policy does not saturate so much on the 9x9 grid—the larger problem space provides more room for improvement. Future work should involve a more efficient implementation, dividing the agents among several machines. We expect that the `best response` algorithm will then scale well as the number of agents is increased.

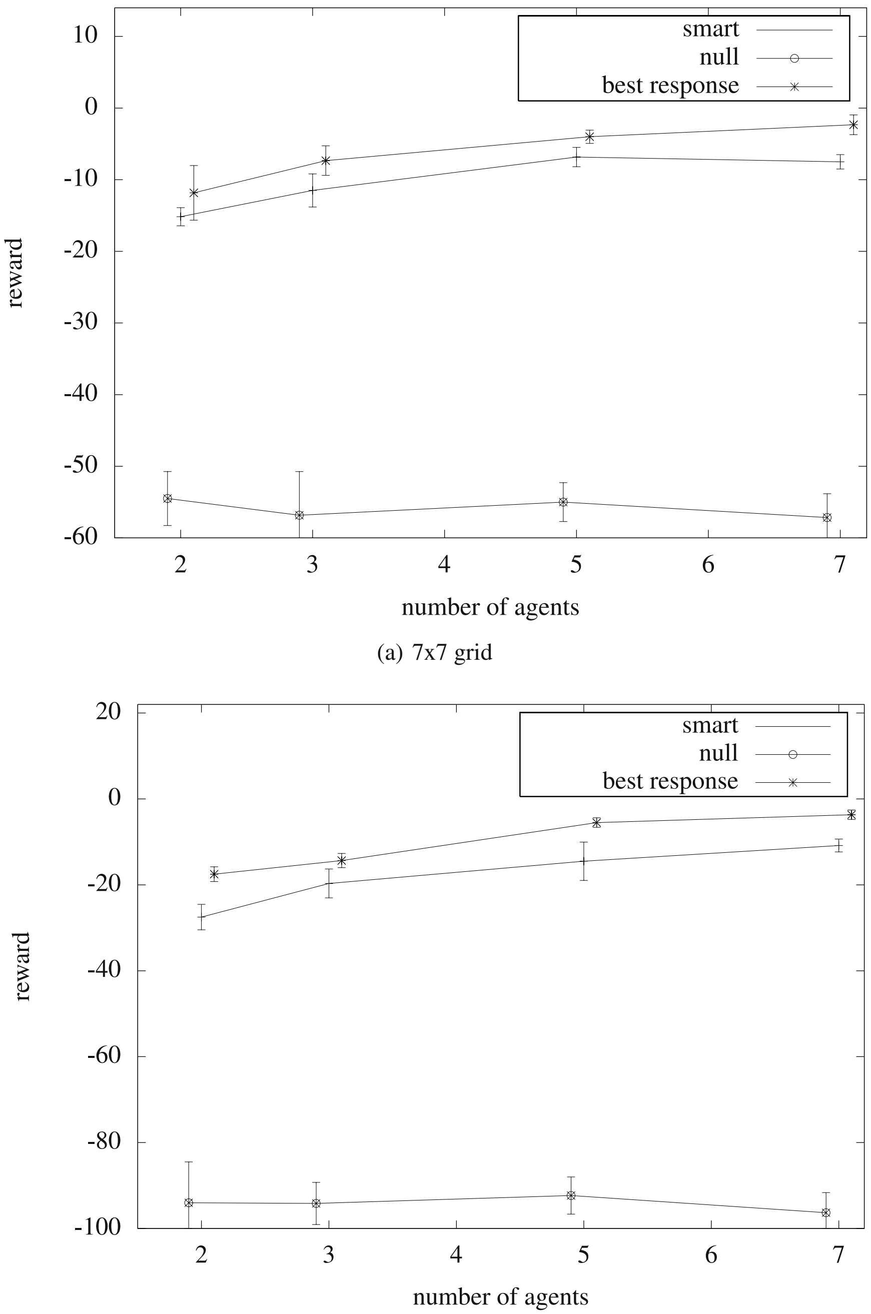
The `best response` algorithm also performs well on the large grids with many millions of states: with five agents nearly all the victims are rescued (the reward does not fall far below 0) even on the largest (12x12) grid. The `smart` policy falls away by comparison. This reflects the results we have seen earlier where the



(a) 2 agents on 3x3 grid

(b) 3 agents on 7x7 grid

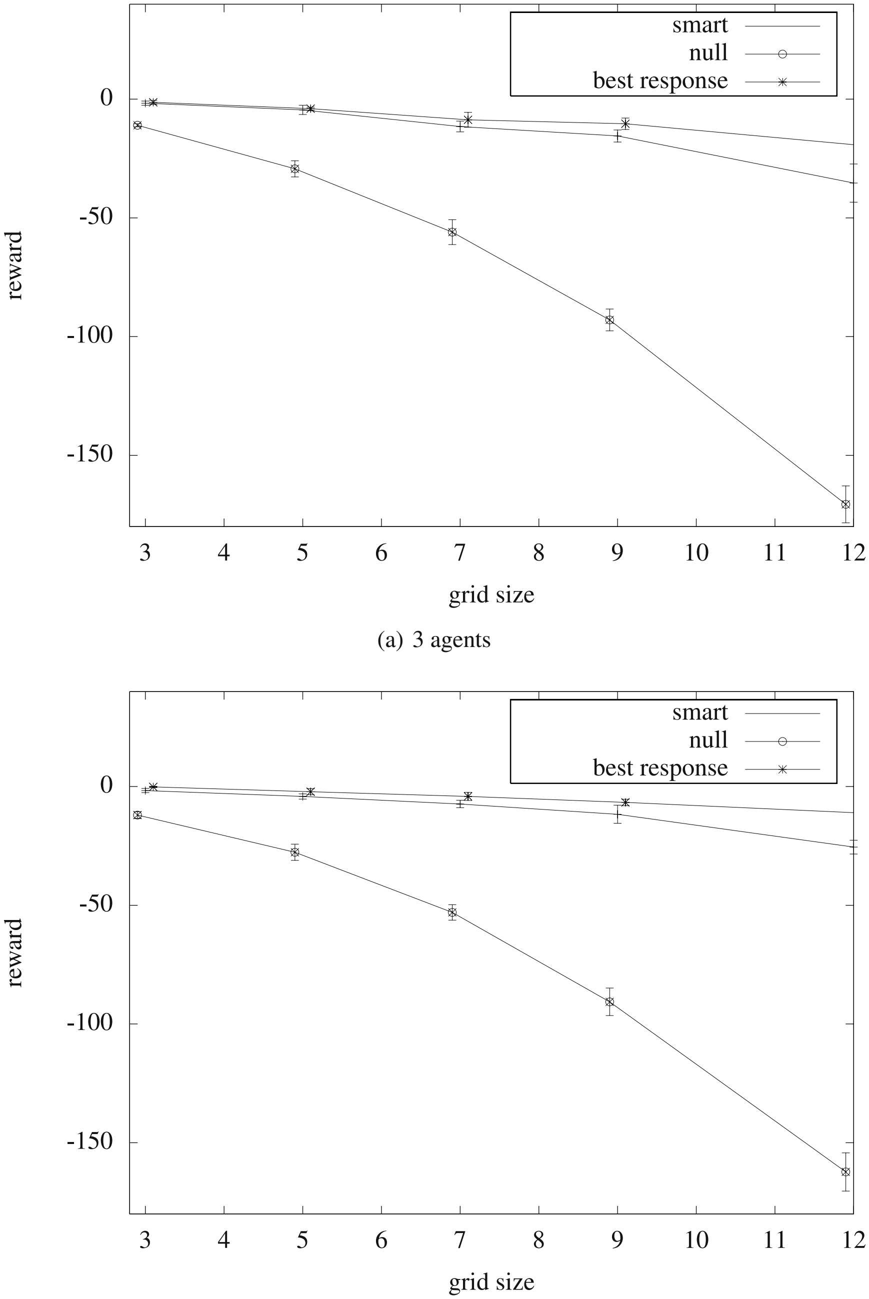
Fig. 10 Effects of varying victim arrival rate



(a) 7x7 grid

(b) 9x9 grid

Fig. 11 Effects of increasing the number of agents on the results for two large grids



(a) 3 agents

(b) 5 agents

Fig. 12 Effects of changing the grid size on the results for 3 and for 5 agents

best response improves over the smart policy more as the grid size increases, a consequence of the way in which the best response policy incorporates uncertainty and the need for search on larger grids. The results are very similar for both three agents (Figure 12(a)) and five agents (Figure 12(b)) although, as expected, five agents are able to make more rescues than three agents (the lines are slightly flatter).

Thus, we have observed that the best response algorithm performs well by comparison with a handwritten strategy designed for the same problem, and requiring much less sampling than the POSG algorithm to achieve this performance. Although the best response algorithm typically has similar performance to the handwritten strategy, it is consistently outperforming it. Furthermore, the best response algorithm scales well, solving problems with many states and increasing numbers of agents and improving on the handwritten strategy for these large problems. Since in general we anticipate our algorithm to be useful in scenarios where no good handwritten strategy is available, especially as the problem scales, the best response algorithm seems promising. Further improvements are discussed in the next section.

7 Conclusions and Future Work

In summary, we have considered the problem of agent coordination in uncertain and partially observable systems. We developed an approach to this problem using a Bayesian learning mechanism, extending previous work on learning models of other agents, and demonstrated its effectiveness on a cooperative scenario from the disaster response domain. To emphasize, the novelties in this work lie in an extension of online model-based learning techniques into partially observable domains, using finite automata. As part of our theory, we outline a general Bayesian model of which our model forms a specific instantiation and show how other techniques, such as POMDPs and Bayesian learning, fit into this same model.

We have examined the performance of our algorithm on a rescue problem with respect to differing problem parameters, finding that its performance consistently outperforms a handwritten strategy for this problem, more noticeably so as the number of agents and the number of states involved in the problem increase. We also observe that reducing the sampling rate of our algorithm has only small effects on its performance, indicating that the best response calculation is the most important feature—this is encouraging, as it enables us to use the best response algorithm with few samples, resulting in greater efficiency. However, we have commented that the limiting factor in running our algorithm, particularly as the number of agents increases, is the memory usage of our implementation, rather than the per-step time required. We therefore propose that future work should investigate more efficient implementations, and ways of distributing the problem across several machines—this is in any case a more accurate model of the problems of interest to us.

Although the work described above is encouraging, there remain a number of areas in which improvement can be made. As well as scaling the model into higher numbers of agents and larger state spaces, using a more efficient implementation for the environment and agents, and running the agents on distributed machines, there are improvements which can be made to the model. We discuss each of these in turn below.

Firstly, we propose to improve upon the learning of the FSM, using automatic state clustering. In the rescue problem, and in many other problems, groups of states can be considered equivalent by the agents. As a simple demonstration, note that there are several symmetries in our example problem: at every step the grid can be rotated until our agent is towards, say, the bottom right, dividing state space into equivalence classes with four states in each class, one corresponding to each rotation $(90^o, 180^o, 270^o, 0^o)$. More generally, we need only as many abstract states as there are joint actions, associating every underlying state with its optimal joint action. However, in practice, particularly if we plan to re-use parts of our model, reducing it purely to joint actions will be too abstract. An appropriate abstraction algorithm should be adaptable, allowing us to change our mind about which action is associated with a particular state, should allow us to update clusters incrementally and should not tie us to any predefined set of clusters. We propose to use a form of statistical clustering based on that described in [18] for this purpose.

A second area of improvement is to better exploit the information available to agents. We are investigating a complex problem domain in which some domain knowledge can be assumed. We may also be able to assume some level of rationality in the other agents (akin to coordination *conventions*). As we develop our models of the agents, we have discussed how we can use these models to improve our beliefs about the agents' observations, applying Bayes' rule. However, it may be possible to make more sophisticated belief updates by considering the observations which we make and the observations which other agents will make to be correlated streams of information. Techniques such as the Kalman Filter [40] are able to operate over correlated streams of information to make more accurate estimates about the value of any particular point and to estimate missing data [25]. These techniques could be applied (with caution) to our estimates of the observations of the other agents and of the current state.

Thirdly, we propose to move beyond the scope of the current work, considering cases in which the environmental dynamics are unknown or are changing, and in which agents are able to enter and leave the environment as the problem progresses. As discussed in Section 3, the algorithm we have presented can in principle be used to learn fixed parameters such as parts of the environmental dynamics, by treating these parameters as a part of a "grand state" from which observations are made. Indeed, related work [8] [30] has done this for some special cases.

In this context, given the uncertainties of our domain, it is clear that if the behaviour of the other agents is completely unknown, **and** the current state is unknown, **and** the environmental parameters are completely unknown, an agent must stumble around "in the dark" for some considerable time before it can begin to get a handle on good or optimal behaviour. However, in the typical scenarios motivated by our example domain of disaster response, an agent will have strong prior information about some or all of the unknown parameters. For example, the other agents may be assumed to be rational and cooperative, thus likely to behave in a near-optimal way. In our example problem, the form of the transition function may be known, but not the exact values of every parameter. By incorporating all the information available to the agent into its model, and particularly by correlating information, we anticipate that our model will be able to handle problems in which the environmental dynamics are not completely known using the theoretical form laid out in Section 3.

Following on from this, our model will easily handle scenarios in which the number of agents changes (but is known to our agent) over time. Since the best response is computed at each step, there will be no difficulty in computing a best response over a subset of the other agents, or in adding a new agent model to the collection. Our agent will adapt continually during the problem run. Similarly, if the agent is learning the environmental dynamics, and those dynamics change, the agent should adjust its model smoothly.

With these improvements, we anticipate that the model of Section 3 can be used as the basis of an algorithm capable of solving medium-sized distributed collaborative problems in the real world, such as traffic management, controlling search robots in a building after a fire, or distributing ambulances during a disaster. Similar algorithms could also be included in software which could be loaded onto handheld devices to aid human decision-making during critical situations such as war or a large-scale disaster.

Acknowledgements. Thanks to Georgios Chalkiadakis and Zinovi Rabovich for discussions on the early versions of this work.

References

1. Aberdeen, D., Baxter, J.: Scaling internal-state policy-gradient methods for POMDPs. In: Proceedings of the 19th International Conference on Machine Learning, vol. 2, pp. 3–10. Morgan Kaufmann, San Francisco (2002)
2. Abul, O., Polat, F., Alhajj, R.: Multiagent reinforcement learning using function approximation. IEEE Transactions on Systems, Man, and Cybernetics, Part C 30, 485–497 (2000)
3. Amato, C., Bernstein, D.S., Zilberstein, S.: Solving POMDPs using quadratically constrained linear programs. In: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pp. 341–343. ACM Press, New York (2006)
4. Boutilier, C.: Planning, learning and coordination in multiagent decision processes. In: Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge, pp. 195–210. Morgan Kaufmann Publishers Inc., San Francisco (1996)
5. Bowling, M., Veloso, M.: Rational and convergent learning in stochastic games. In: International Joint Conferences on Artificial Intelligence, pp. 1021–1026 (2001)
6. Carmel, D., Markovitch, S.: Learning models of intelligent agents. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, Oregon, vol. 2, pp. 62–67 (1996)
7. Cassandra, A., Littman, M., Zhang, N.: Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In: Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence, pp. 54–61. Morgan Kaufmann, San Francisco (1997)
8. Chalkiadakis, G., Boutilier, C.: Coordination in multiagent reinforcement learning: a Bayesian approach. In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pp. 709–716. ACM Press, New York (2003)
9. Clark, A., Thollard, F.: PAC-learnability of probabilistic deterministic finite state automata. Journal of Machine Learning Research 5, 473–497 (2004)
10. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence, Menlo Park, CA, USA, pp. 746–752. American Association for Artificial Intelligence (1998)

11. Dearden, R., Friedman, N., Andre, D.: Model-based Bayesian exploration. In: Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence, pp. 150–159. Morgan Kaufmann, San Francisco (1999)
12. Durfee, E.H.: Practically coordinating. AI Magazine 20(1), 99–116 (1999)
13. Dutta, P.S., Dasmahapatra, S., Gunn, S.R., Jennings, N., Moreau, L.: Cooperative information sharing to improve distributed learning. In: Proceedings of the AAMAS 2004 workshop on Learning and Evolution in Agent-Based Systems, pp. 18–23 (2004)
14. Emery-Montemerlo, R., Gordon, G., Schneider, J., Thrun, S.: Approximate solutions for partially observable stochastic games with common payoffs. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, Washington, DC, USA, pp. 136–143. IEEE Computer Society, Los Alamitos (2004)
15. Fischer, F., Rovatsos, M., Weiss, G.: Hierarchical reinforcement learning in communication-mediated multiagent coordination. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, Washington, DC, USA, pp. 1334–1335. IEEE Computer Society, Los Alamitos (2004)
16. Fitoussi, D., Tennenholtz, M.: Choosing social laws for multi-agent systems: Minimality and simplicity. Artificial Intelligence 119(1-2), 61–101 (2000)
17. Fudenberg, D., Levine, D.K.: The Theory of Learning in Games. MIT Press, Cambridge (1998)
18. Hoar, J.: Reinforcement learning applied to a real robot task. DAI MSc Dissertion, University of Edinburgh (1996)
19. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101(1-2), 99–134 (1998)
20. Kim, Y., Nair, R., Varakantham, P., Tambe, M., Yokoo, M.: Exploiting locality of interaction in networked distributed pomdps. In: Proceedings of the AAAI Spring Symposium on Distributed Plan and Schedule Management (2006)
21. Leslie, D.: Reinforcement learning in games. PhD thesis, University of Bristol (2004)
22. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of the 11th International Conference on Machine Learning, New Brunswick, NJ, pp. 157–163. Morgan Kaufmann, San Francisco (1994)
23. Marecki, J., Gupta, T., Varakantham, P., Tambe, M.: Not all agents are equal: scaling up distributed POMDPs for agent networks. In: Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (2008)
24. National Research Council. Summary of a Workshop on Using Information Technology to enhance Disaster Management. National Academies Press (2005)
25. Osborne, M.A., Rogers, A., Ramchurn, S., Roberts, S.J., Jennings, N.R.: Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. In: International Conference on Information Processing in Sensor Networks, April 2008, pp. 109–120 (2008)
26. Paquet, S., Tobin, L., Chaib-draa, B.: An online POMDP algorithm for complex multiagent environments. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, pp. 970–977. ACM Press, New York (2005)
27. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for POMDPs. In: International Joint Conference on Artificial Intelligence, August 2003, pp. 1025–1032 (2003)
28. Poupart, P., Vlassis, N., Hoey, J., Regan, K.: An analytic solution to discrete bayesian reinforcement learning. In: Proceedings of the 23rd international conference on Machine learning, pp. 697–704. ACM, New York (2006)
29. Ramamritham, K., Stankovic, J.A., Zhao, W.: Distributed scheduling of tasks with deadlines and resource requirements. IEEE Transactions on Compututers 38(8), 1110–1123 (1989)
30. Ross, S., Chaib-draa, B., Pineau, J.: Bayes-adaptive POMDPs. In: Neural Information Processing Systems (2008) (in press)

31. Roy, N., Gordon, G.: Exponential family PCA for belief compression in POMDPs. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing, Vancouver, Canada, December 2002, pp. 1043–1049 (2002)
32. Scerri, P., Liao, E., Xu, Y., Lewis, M., Lai, G., Sycara, K.: Coordinating very large groups of wide area search munitions. In: Theory and Algorithms for Cooperative Systems (2005)
33. Scerri, P., Sycara, K., Tambe, M.: Adjustable autonomy in the context of coordination. In: AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit (2004) (invited paper)
34. Shani, G., Brafman, R.I., Shimony, S.E.: Model-based online learning of POMDPs. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS, vol. 3720, pp. 353–364. Springer, Heidelberg (2005)
35. Smith, A.J.: Dynamic generalisation of continuous action spaces in reinforcement learning: A neurally inspired approach, Ph.D. thesis, Division of Informatics, Edinburgh University, UK (2002)
36. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
37. Tambe, M., Adibi, J., Alonaizon, Y., Erdem, A., Kaminka, G.A., Marsella, S., Muslea, I.: Building agent teams using an explicit teamwork model and learning. Artificial Intelligence 110(2), 215–239 (1999)
38. Vu, T., Powers, R., Shoham, Y.: Learning against multiple opponents. In: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pp. 752–759. ACM, New York (2006)
39. Wang, F.: Self-organising communities formed by middle agents. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems, pp. 1333–1339. ACM Press, New York (2002)
40. Welch, G., Bishop, G.: An introduction to the Kalman filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA (1995)
41. Wooldridge, M.: An Introduction to Multi-agent Systems. Wiley, Chichester (2002)

Collaborative Agents for Complex Problems Solving

Minjie Zhang, Quan Bai, Fenghui Ren, and John Fulcher

Abstract. Multi-Agent Systems (MAS) are particularly well suited to complex problem solving, whether the MAS comprises cooperative or competitive (self-interested) agents. In this context we discuss both dynamic team formation among the former, as well as partner selection strategies with the latter type of agent. One-shot, long-term, and (fuzzy-based) flexible formation strategies are compared and contrasted, and experiments described which compare these strategies along dimensions of Agent Search Time and Award Distribution Situation. We find that the flexible formation strategy is best suited to self-interested agents in open, dynamic environments. Agent negotiation among competitive agents is also discussed, in the context of collaborative problem solving. We present a modification to Zhang's Dual Concern Model which enables agents to make reasonable estimates of potential partner behavior during negotiation. Lastly, we introduce a Quadratic Regression approach to partner behavior analysis/estimation, which overcomes some of the limitations of Machine Learning-based approaches.

1 Introduction

Complex problem solving typically requires diverse expertise and multiple techniques. Over the last few years, Multi-Agent Systems (MASs) have come to be perceived as a crucial technology, not only for effectively exploiting the increasing availability of diverse, heterogeneous, and distributed on-line information resources, but also as a framework for building large, complex, and robust distributed information processing systems which exploit the efficiencies of organized behaviour. MAS technology is particularly applicable to complex problem solving in many application domains, such as distributed information retrieval [22], traffic monitoring systems [32], and Grid computing [35], etc.

Minjie Zhang, Quan Bai, Fenghui Ren, and John Fulcher
School of Computer Science and Software Engineering, University of Wollongong,
Wollongong NSW 2522, Australia
e-mail: {minjie,quan,fr510,john}@uow.edu.au

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 361–399.
springerlink.com

A MAS comprises a group of agents, which can collaborate when dealing with complex problems, or alternatively perform tasks individually with high autonomy. In a MAS, agents can be characterised as either ‘self-interested’ or ‘cooperative’ [21] [34]. When different types of agents work together, management of their interactions is a very important and challenging issue for the success of MASs.

This chapter introduces two main approaches for complex problem solving via agent cooperation and/or competition, these being (i) a partner selection strategy among competitive agents, and (ii) dynamic team forming strategies among cooperative agents.

This chapter is organised as follows. Section 2 provides some background knowledge and definitions relevant to agents and MASs. In Section 3, a dynamic team-forming approach for MASs in open environments is introduced, which can be used among both cooperative and self-interested agents. In Section 4, a fuzzy logic approach for partner selection among self-interested agents via agent competition is discussed in detail. The chapter concludes and further research outlined in Section 5.

2 Self-interested and Cooperative Multi-Agent Systems

2.1 Traditional Classification

Agent activities are driven by their goal(s), and according to the properties of these goal(s), can be classified as either ‘self-interested’ (competitive) or ‘cooperative’ (benevolent) agents [20] [21]. These two types of MAS can be defined as follows:

Definition 1. A MAS that contains agents with distinct or even competitive individual goals is defined as a self-interested MAS.

Generally, an agent of a self-interested MAS collaborates with other agents to realise or maximise their local utilities.

Definition 2. A MAS that contains agents with common goals is defined as a cooperative MAS.

Normally, agents of a cooperative MAS work together toward maximising the realization of their common goal(s).

An example of a cooperative MAS application is RoboCup [4] [5] [18]. In a robot soccer team, all robot players (agents) collaborate to achieve their common goal, i.e., winning the game. A typical example of a self-interested MAS is an agent-based e-Commerce system in an electronic marketplace [15] [23] [39] [40]. In an electronic marketplace, different agents work in the same environment toward non-cooperative individual goals. However, agents still need to collaborate with others in order to maximise their individual utilities, i.e. purchase/sell items collaboratively in order to obtain the best price(s).

2.2 The Blurred Boundary

As the sophistication of MASs increases, the traditional classification of 'self-interested' versus 'cooperative' MAS becomes impractical and unreasonable in many domains [42]. In many MAS applications, a MAS can neither be a simple market system nor an agent colony. The boundary between self-interested and cooperative MASs thus becomes blurred [20] [42]. This is mainly due to the following reasons:

1. In many current MAS applications, agents can come from different organisational entities. These agents work together because the organisations they belong to have some cooperative relationships [27]. Therefore the terms and conditions of this cooperation between individual agents mainly depend on the higher-level relationship between the organisations. This kind of MAS is not purely self-interested because of the existence of common goal(s) among the agents. However such MASs can neither be classified as typically cooperative because cooperation between agent members are facile and depend not only on the system's overall utility but also on many outside factors.
2. In many MAS applications, self-interested agents are also required to take care of the global system utility via temporal cooperation in order to maintain and improve their working environments. As the social welfare of the system increases, all system members, including self-interested agents, will benefit.
3. A MAS can include agents from different organisational entities. This leads to an agent in the MAS having different attitudes toward different targets. An agent can be cooperative with agents from the same organisation as itself, yet act in a self-interested manner with agents of other organisations. Therefore, a MAS could be a system comprising both self-interested and cooperative agents. In this situation, it is difficult to identify whether the MAS is cooperative or self-interested.
4. Even within the same organisation, cooperative agents may also behave in a self-interested way due to their limited local view [16] [42].

2.3 Two Scenarios

In Sections 3 and 4, we introduce first a team-forming mechanism for cooperative problem solving via agent cooperation, followed by a partner section approach for collaboration via agent competition, in various types of agent systems.

The following two scenarios will be used in Section 3 and Section 4, respectively, to demonstrate the application of our proposed approaches supporting by experimental results.

Scenario 1
In a general service composition system, a number of services need to be combined together to execute a task in the system. For instance, if we want to transport goods overseas, we have to combine several kinds of services together, which might include packing service, road transport service, custom elated service and shipping service. An agent in a service composition system is normally used to represent a

particular service, and the resource of the agent is the service that the agent can provide. In such a system, agents must work with each other like a team in order to achieve the desired goal i.e. to execute tasks cooperatively because each task must be accomplished by more than one services.

Scenario 2
A car buyer wants to purchase a car. However, there are several prospective sellers. To avoid extensive negotiation with each seller, the buyer should filter out some 'impossible' car sellers. For example, if a car seller's bid is much higher than the buyer's expectation or the seller's reputation cannot be trusted by the buyer, then the buyer will filter out such car sellers by employing the partner selection approach before the negotiation starts. During the negotiation, in order to maximise self's profit, the car buyer can predict its negotiation partner's behaviors and make corresponding responses. For example, for a car buyer in a hurry, if he estimates that a car seller cannot make further concession, then he will not spend more time on the current bargaining but looks for another possible seller. On the other hand, for a patient car buyer, if he estimates that a car seller still has scope to make future concessions, then the car buyer will make more effort on the bargaining. Therefore, by employing the behaviours prediction approach, the agent can get some advantages in bargaining.

In distributed and complex problem solving, many MAS applications face a similar situation as Scenario 1, such as Web-based grid computing, distributed information gathering, distributed monitoring systems, automated design and production lines. Scenario 2 is a typical example for self-interested MASs in the domain of e-commerce and frequently happens in wide agent-based e-trading and e-market places. Section 3 and Section 4 introduce the detail definitions and principles about two proposed approaches for agent collaboration, and also demonstrate experimental results about how to achieve agent collaboration through dynamic team formation in Scenario 1, and how to achieve agent collaboration by using a partner selection strategy in Scenario 2, respectively.

3 Collaborative Problem Solving through Agent Cooperation

As introduced in the previous section, MASs can be classified as either self-interested or cooperative, according to the features of agent goals. However, cooperation is unavoidable in most MASs regardless of whether or not they are cooperative or self-interested. Due to the distributed nature of the problem to be solved, and because of limitations in agent abilities, in many cases agents need to work together on some tasks (i.e. via cooperation).

Agent abilities are limited. To perform tasks beyond its inherent ability, an agent needs to collaborate with other agents through joining or forming a particular organisation. The organisation of a group of agents is the collection of roles, relationships and authority structures which govern agent behaviours [14]. All MASs possess some form of organisation to support agent interactions. The form of organisation guides how the agent members interact with each other. An agent team is a kind of organisational structure that supports agent cooperation. Generally speaking, each agent team is composed of a team leader and several team members. After an agent joins a team, it will cooperate with other team members towards a common goal.

In current MAS research, MAS team formation is faced with a number of challenges, especially with regard to the following two aspects:

- Many current multi-agent systems (MASs) are required to work in open and dynamic environments [1] [13] [37] [38]. Uncertainties of dynamic environments obstruct coherent teamwork and bring difficulties for agent cooperation. In dynamic environments, system constraints, resource availability, agent goals, etc. are all changeable. Changing any of these factors may directly require a MAS to deal with different situations. In a new situation, retaining outdated cooperative relationships may obstruct agents in achieving their individual goals.
- Compared with cooperative agents, cooperation among self-interested agents is more complicated and dynamic, due to their selfish features. Self-interested agents are impelled to cooperate with others by their individual goals (due to limited individual abilities). In an agent team composed of self-interested agents, temporary cooperation among agents might conflict with the selfish goals of individual agents as the environment changes. In open and dynamic environments, if factors such as agent goals, task requirements and resources change, a selfish agent may need to modify or even terminate the cooperative relationships with its colleagues, otherwise the cooperation would be in conflict or even be harmful to the individual agent goal. Considering this point, some researchers suggest using dynamic agent cooperation strategies in this kind of application. However, how long cooperation should be maintained among particular agents is always a problem.

In many MAS applications, a dynamic team-formation mechanism is needed to enable agents to automatically form and reform groups/teams to avoid profit conflicts between agents in line with changes in the environment. Toward this objective, a number of researchers try to find an optimal mechanism for dynamic team formation and member selection [30] [36] [37] [38]. Generally, in current MAS research, there are two kinds of team-formation mechanisms in widespread use, these being one-shot team formation and long-term team formation. These team-formation mechanisms are described below:

- *One-shot team-formation mechanism (for temporal cooperation)*
  In self-interested MASs, an individual agent's willingness and goals are important factors that need to be considered during team formation. Research on team formation for self-interested agents generally focuses on forming one-shot teams, also called short-term teams, for individual tasks. In this kind of mechanism, agents come together when they need to handle some tasks, and their relationships will be terminated after the tasks have been accomplished.
- *Long-term team-formation mechanism (for long-term cooperation)*
  Obviously, one-shot teams can experience frequent grouping and regrouping among agents. Each grouping/regrouping consumes some communication and

computation resources. To overcome the weakness of one-shot team formation, Rathod and desJardins proposed several stable-team formation strategies for self-interested MASs [30]. These strategies allow self-interested agents to form long-term relationships in order to reduce team formation overhead. However, for many self-interested MASs, agent goals or willingness are changeable and remain uncertain. A long-term relationship is very difficult to maintain after the goals of team member agents change.

Both one-shot team formation and long-term team-formation mechanisms have some weaknesses. One-shot team formation may bring high communication and computation overhead to a MAS. However, long-term team formation strategies are not suitable for the dynamic features of open environments and the selfish features of self-interested agents.

In this section, we introduce and compare the features of the one-shot and long-term team-formation mechanisms. In addition, to cover some shortcomings of one-shot and long-term team formations, a flexible team-formation mechanism that enables both cooperative and self-interested agents to flexibly choose team membership and duration is proposed. Factors such as historical agent performance, task requirements and resource constraints are considered in the mechanism. Especially for open environments, flexible team formation and member selection mechanisms are more suitable for agent applications. This flexible team-formation mechanism enables more dynamic and reasonable cooperation between agents and reduces unnecessary overhead and utility conflicts brought about by team formation. Due to the high uncertainty inherent in most open environments, analysis and evaluation of dynamic factors is not very straightforward. More specifically, a fixed standard for agent evaluations does not exist (e.g. how good an agent's performance is). Regarding this point, fuzzy rules are used in our flexible team-formation mechanism to evaluate the performance and importance of agents. This will enable an agent to dynamically select cooperation durations and objectives based on the results of fuzzy evaluations, and to choose cooperation mechanisms more flexibly.

3.1 Agent Cooperation in Agent Teams: The Scenario

Various MAS applications may have different system structures. In this chapter, an MAS environment is set up to demonstrate and analyse team formation and member selection mechanisms. Hence, the system structure is set up toward assisting agent communication and task allocation. Some simplifying assumptions and definitions, which can avoid adding to the scheduling and task decomposition problems, are also made, and only elementary agents and task models are included in the MAS. However, these models are sufficiently generic to be practical and applicable to a wide range of real-world applications.

Figure 1 shows the general structure of team organisation. To simplify the problem, we assume that all agents are aiming to achieve rewards through accomplishing tasks sent by outside users. New tasks are published on the system *Task Board*, and will be removed from the *Task Board* after being taken by an agent or agent team.

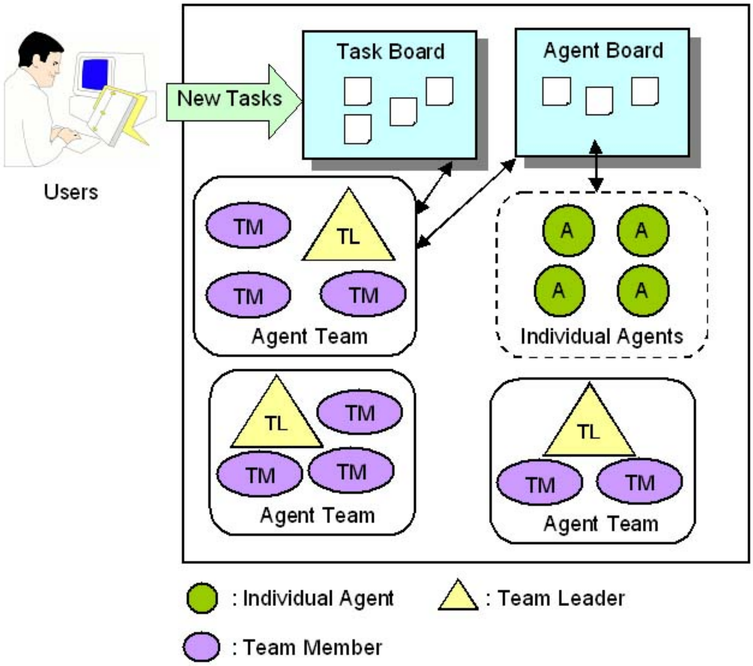


Fig. 1 The System Architecture

Published tasks are accessible to all individual agents and agent teams within the system. The number of agents in the system can be dynamic; agents can enter and leave the system at will. However, agents need to publish and remove their registration information on the system *Agent Board* before they so enter (leave). The registration information records the skills and status of an agent (see Definition 4).

Agent abilities are limited. To perform tasks beyond their individual ability, an agent needs to collaborate with other agents through joining or forming a team. Each agent team is composed of one (and only one) team leader and several team members. After an agent joins an agent team, it can receive payments from the agent team. At the same time it needs to work for the agent team for a certain period. The payment and serving term are described in the contract (see Definition 5) between the team member and the team leader.

Before presenting the team-formation mechanism, some important definitions and assumptions are given.

Definition 3. A *task* is defined formally as $t_i = \langle w_i, R'_i \rangle$, where w_i is the reward gained by an agent/agent team if task t_i is accomplished by that agent/agent team; R_i is the set of resources or skills, which are possessed by agents, required by task t_i. A task can only be assigned to one agent or agent team.

Definition 4. An *agent* is formally defined as $a_i = \langle g_i, R_i, s_i \rangle$, where g_i is a set of individual goals of agent a_i; R_i is the skills and resources possessed by agent

Table 1 Status of An Agent

s_i **value**	**Status of agent** a_i
(0, 0, 0)	Performing no task; has no agent team.
(1, 0, 0)	Performing a task; has no agent team.
(0, 1, 0)	Has a one-shot contract as a team member; performing no task currently.
(1, 1, 0)	Has a one-shot contract as a team member; performing a task currently.
(0, 1, t)	Team member of an agent team for period t; performing no task currently.
(1, 1, t)	Team member of an agent team for period t, performing a task currently.
(0, 2, 0)	The team leader of an agent team; performing no task currently. (It is assumed that the team leader cannot quit from its agent team and let t value of a team leader equal to 0.)
(1, 2, 0)	The team leader of an agent team; performing a task currently.

a_i; and s_i is the status of a_i, where $s_i = (v_a, v_p, t)$. s_i represents whether agent a_i is performing a task and participating in an agent team. The meanings of different s_i values are listed in Table 1. The names and meanings of v_a, v_p and t are as follows:

Availability v_a: represents whether an agent is performing a task. $v_a = 0$ when the agent has no task (available); $v_a = 1$ when the agent is performing a task (not available);

Position Parameter v_p: represents whether an agent is an individual agent, team leader or team member. $v_p = 0$ when the agent is an individual agent; $v_p = 1$ when the agent is a team member; $v_p = 2$ when the agent is a team leader.

Contract Completion Time t: t is the contract completion time of an agent (also see Definition 5).

Definition 5. A *contract* c_{ij} is an agreement between team leader a_i and team member a_j. It can be defined as $c_{ij} = \langle t_{ij}, p_{ij}, S_{ij} \rangle$, where t_{ij} is the contract completion time; p_{ij} is the penalty that the team leader or team member has to pay (to the other parties of the contract) if it breaks the contract and terminates the cooperation relationship before t_{ij}; S_{ij} is a set of payments that a_j can gain through serving the agent team. S_{ij} can be described as a tuple $\langle sc_{ij}, sd_{ij} \rangle$. For contracts between the team leader and team members of a one-shot team, t_{ij}, p_{ij}, and sd_{ij} are equal to 0. sc_{ij} is the payment that a_j can gain for each task completed by the agent team when a_j directly participates in the task. sd_{ij} is the dividend (or reward) that a_j can share for each task completed by the agent team, when a_j does not actually participate in that task.

Definition 6. An *agent team* is a set of agents. It can be formally defined as $AT_i = \langle MS_i, TR_i \rangle$, where MS_i is the set of agents (including the team leader) that are currently team members of AT_i; TR_i is the total resources of the entire agent team. Here it is assumed that $TR_i = \sum_{j|a_j \in MS_i}({}_j) + r_i$, where R_i and R_j are resources possessed by the team leader and team members, respectively. In other words, the capability of an agent team is the sum of its team members' capabilities and the team leader's capability. We further define $\forall i \neq j : MS_i \cap MS_j = \varnothing$, which means that an agent can only participate in a single agent team.

Definition 7. A *Contributor Set* $CS_{ij}(CS_{ij} \subset MS_i)$ of agent team AT_i is the set of agents that participate in performing task t_j, where t_j is a task of AT_i. For a one-shot team, the Contributor Set is equal to MS_i of the team (also refer to Definition 6).

Definition 8. For agent team AT_i, a *Member Contribution* mc_{ijk} is the contribution of agent a_k, where $a_k \in CS_{ij}$, in performing task t_j ($t_i = \langle w, R'_i \rangle$). mc_{ijk} equals w/N, where N is the size of Contributor Set and w is the task reward.

3.2 One-Shot and Long-Term Team-Formation Mechanisms

After presenting the system architecture and some important definitions, concepts and comparisons of the one-shot and long-term team-formation mechanisms are presented in this subsection.

3.2.1 One-Shot Team-Formation Mechanism

One-shot team-formation mechanisms are widely applied in many MAS applications. In this mechanism, agents do not initially have a team. When a task t_i is published in the *Task Board*, agents start to bid on the new task. The system facilitator will choose (or randomly select) a bidder to assign the task. After the agent successfully bids for the task, it becomes a team leader and starts to look for collaborators according to the task requirement R'_i. Finally, the agent team will disband after the task (t_i) is accomplished.

Generally, the one-shot team strategy includes the following processes. (Here, it is assumed that the agents cannot achieve the task individually.)

1. The system facilitator of the MAS publishes a new task $t_i = \langle w_i, R'_i \rangle$ on the *Task Board*, where w_i and R'_i are the reward and required resources of the task;
2. Agents, whose $g < w_i$ and s=(0, 0, 0) bid on t_i;
3. The system facilitator awards t_i to agent $a_j (a_j =< g_j, R_j, s_j >)$. At the same time, a_j becomes the team leader of agent team AT_j and modifies its s_j to (0, 2, 0). At this movement, $TR_j = R_j$;
4. a_j searches the Agent Board to look for agents with status (0, 0, 0), which can provide the lacking resources R, where $R \subseteq (R'_i - R'_i \cap TR_j)$;
5. a_j finds a required agent a_p, where $R_p \subseteq (R'_i - R'_i \cap TR_j)$;
6. a_j sends a contract c_{jp} to a_p, where $sc_{jp} \leq (w_i - g_j) \cdot sizeOf(R_p)/sizeOf(R'_i - R_i)$;
7. a_p accepts c_{jp} if $sc_{jp} \geq g_p$ or rejects c_{jp} if $sc_jp \leq g_p$;
8. If c_{jp} is accepted by a_p, $TR_j = TR_j \cup R_p$, and a_p modifies its status to (0, 1, 0);
9. Goes to Process (4) until $TR_j = R'_i$;

10. AT_j starts to perform t_i; the team leader and the team members of AT_j modifies/modify its/their statuses to (1, 1, 0) and (1, 2, 0), respectively;
11. AT_j accomplishes t_i; agents of AT_j modify their statuses to (0, 0, 0) and are released from the team.

3.2.2 Long-Term Team-Formation Mechanism

In the long-term team-formation mechanism, the agent team will not be dissolved after performing tasks. On the contrary, the team leader gives the team members some payment to maintain the cooperative relationship, even if the team member does not contribute to accomplishing the task.

The long-term team strategy normally includes the following processes [30]:

1. Team leader a_i finds several free agents, whose status values are (0, 0, 0), from the *Agent Board* and sends them contracts in order to form a team with them. Agents modify their status to (0, 1, t_{ij}) if they accept the contracts. In this case, agent team AT_i is formed successfully;
2. Team leader a_i searches the *Task Board* for a suitable task and bids on task $t_k(t_k = \langle w_k, R'_k \rangle)$, where $R'_k \subseteq TR_i$ and $w_k \geq \sum_{j|a_j \in MS_i}(S_{ij} + g_i)$ (also refer to Definitions 3 through 6).
3. If t_k is successfully bid by team leader a_i, a_i assigns t_k to team member $a_p, a_q...a_n$, where $R_p \cup R_q, ..., \cup R_n$ is the minimum set that satisfies $R'_k \subseteq R_p \cup R_q, ..., \cup R_n$. At the same time, $a_p, a_q, ..., a_n$ modify their status to $(1,1,t_{ip})$, $(1,1,t_{iq}), ..., (1,1,t_{in})$. Also, for this task performance, the Contributor Set CS_{ik} (refer to Definition 7) should be $\{a_p, a_q, ..., a_n\}$;
4. $a_p, a_q, ..., a_n$ modify their status to $(0,1,t_{ip}), (0,1,t_{iq}), ..., (0,1,t_{in})$ after t_k is accomplished;
5. team leader a_i awards team member a_m ($a_m \in AT_i$) with $(sc_{im} + sd_{im})$ if $a_m \in CS_{ik}$, or sd_{im} if a_m is not in CS_{ik};

In addition, if the team leader a_i or team member a_p wants to terminate the contract before the contract completion time t_{ip}, they may process the following two steps:

1. a_i/a_p terminates c_{ip} with a_p/a_i, and pays p_{ip} to a_p/a_i;
2. a_p is released from AT_i, and its status modified to (0, 0, 0).

3.2.3 Advantages and Disadvantages of Long-Term and One-Shot Team-Formation Mechanisms

One-shot teams are suitable for dynamic MAS application domains. They always maintain loosely-coupled relationships among agents by default. However, agents in dynamic applications may also need to keep stable organisations in some situations. For example, the tasks may have some similarity, and their requirements might be similar (which means they may just need similar agent teams). In this case, frequent grouping and regrouping is not necessary, since each such grouping consumes some

Table 2 Features of One-Shot Teams and Long-Term Teams

	One-Shot Teams	**Long-Term Teams**
Communication Overhead	High	Low
Suitable Domains	Highly dynamic environments	Stable environments
Suitable MASs	Self-interested MAS	Cooperative MAS
Relationships among Team Members	Loosely coupled	Tightly coupled

system resources. In contrast with one-shot teams, long-term teams can greatly reduce the system overhead caused by grouping and regrouping. However, most current long-term team formation strategies cannot figure out when agents should form long-term teams, which agents should be included, and how long the relationships should be maintained. For self-interested MAS applications, keeping unnecessary long-term cooperative relationships could be dangerous and harmful for the overall system performance. Features of one-shot and long-term teams are summarised and compared in Table 2.

3.3 *Flexible Team-Formation Mechanism*

From the description of short-term and long-term team formation in the previous section, it can be seen that both long-term and one-shot teams have some advantages and disadvantages. One-shot teams are suitable for dynamic tasks, where the requirements of various new tasks are totally different. By contrast, long-term teams possess advantages when tasks are 'stable' or similar. For most self-interested agents, the team duration should not be fixed. Taking human society as an example, a company may sign different contracts (with different durations and conditions) with different employees. According to the performance of employees and changes in the job market, the company will typically want to make changes to these contracts in the future. For a MAS, it is also necessary to have a flexible team-formation mechanism which can enable team leaders to choose different cooperation durations with agents, according to the changing trends of task-requirements and agent performance. In this section, a flexible team-formation mechanism is introduced. In this mechanism, agent value and availability are evaluated. Team leaders will then determine the required members and choose proper cooperation durations and cost according to these evaluation results.

3.3.1 Team Member Performance Evaluation

In general, agents that always contribute to performing tasks and can bring more benefits to the team are the most valuable members of an agent team. These agents should be kept on the team for a long time. By contrast, an agent team should not include agents that bring little contribution to the team. In this mechanism, two factors, namely *Utilisation Ratio* (*ur*) and *Contribution Ratio* (*cr*), are used to evaluate the value of a team member.

Definition 9. *Utilisation Ratio* ur_{Mk} ($ur_{Mk} \in [0,1]$) is the frequency with which a team member a_k has participated in the most recent M tasks of the agent team AT_i. It can be calculated using Equation 1. The value of the parameter M is chosen by team leaders or assigned by users. Team leaders can also adjust M values according to environmental situations and team performance.

$$ur_{Mk} = \sum_{j=1}^{M} \frac{1}{M} \quad (\, k|a_k \in CS_{ij}) \tag{1}$$

Definition 10. *Contribution Ratio* cr_{Mk} ($cr_{Mk} \in [0,1]$) is the ratio that team member a_k has contributed to the agent team AT_i in the most recent M tasks. It can be calculated using Equation 2 (also refer to Definition 8).

$$cr_{Mk} = \frac{\sum_{j=1}^{M} mc_{ijk} \quad (\, k|a_k \in CS_{ij})}{\sum_{j=1}^{M} w_j} \tag{2}$$

The following example shows how to evaluate team members through Utilisation Ratio and Contribution Ratio. Suppose $t_1 =< 40, R'_1 >, t2 =< 50, R'_2 >$ and $t_3 =< 60, R'_3 >$ are the three most recent tasks accomplished by agent team AT_i. a_p, a_q, a_r and a_s are the team members of AT_i. Team members that participate in the three tasks are $\{a_p, a_q\}, \{a_p, a_r\}$ and $\{a_p, a_q\}$, respectively. According to Equations 1 and 2 , the Utilisation Ratio and Contribution Ratio values of a_p, a_q, a_r and a_s are:

$$a_p\text{: } ur_{3p} = 1, \quad cr_{3p} = \frac{(40/2+50/2+60/3)}{(40+50+60)} = 0.5$$

$$a_q\text{: } ur_{3q} = 0.67, cr_{3q} = \frac{(40/2+60/3)}{(40+50+60)} = 0.33$$

$$a_r\text{: } ur_{3r} = 0.33, cr_{3r} = \frac{50/2}{(40+50+60)} = 0.17$$

$$a_s\text{: } ur_{3s} = 0, \quad cr_{3p} = 0$$

Comparing Utilisation Ratio and Contribution Ratio values of the four team members of AT_i, it can be seen that a_p is the most important member of AT_i, since it frequently participated in recent tasks and contributed the most benefit to the team. On the other hand, a_s did not participate in recent tasks and contributes nothing to AT_i.

3.3.2 System Agent Resource Evaluation

With Utilisation Ratio and Contribution Ratio, a team leader can evaluate contributions of team members. However, to make reasonable contracts with team members, a team leader also needs to evaluate whether it is easy to find similar agents (which possess similar resources and skills) in the MAS. In this mechanism, Agent

Resource Availability is the parameter defined to evaluate agent resource availability in the MAS.

Definition 11. Agent Resource Availability ara_k: ara_k is the ratio of available agents (which do not have a team/task) that possess the same or more resources than team member a_k. It can be calculated using Equation 3 (Note: N_{av} here is the available agent number of the MAS).

$$ara_k = \sum_{s_i=(0,0,0)}^{R_k \subseteq R_i} \frac{1}{N_{av}} \tag{3}$$

For example, suppose that a_k is a team member of AT_i. Currently, there are ten out of twenty available agents in the MAS, which possess the same or more resources than a_k. Hence, the Agent Resource Availability value of team member a_k is: $ara_k = 0.5$.

3.3.3 Flexible Member Selection Using Fuzzy Rules

According to the values of Utilisation Ratio, Contribution Ratio and Agent Resource Availability, in this mechanism, team leaders use a fuzzy method to determine cooperation durations and cost with their team members.

Input and Output Parameters:

In the fuzzy method, Utilisation Ratio, Contribution Ratio and Agent Resource Availability are input parameters. The output parameters are Contract Term *ct* and Commission Amount *ca*. These parameters are defined in Definitions 12 and 13.

Definition 12. *Contract Term* ct_k is the parameter which denotes the duration that agent a_k should be kept in the agent team. It is an output parameter that needs to be identified through the fuzzy method. The working range of *Contract Term* is [0, *MAXTERM*]. *MAXTERM*, which is a constant that is defined in the MAS, and denotes the maximum time period that an agent can be kept in an agent team.

Definition 13. *Commission Amount* ca_k is the parameter that denotes the maximum commission that the agent team should pay to agent a_k in order to keep it in the team. It is an output parameter that needs to be identified through the fuzzy method. The working range of Commission Amount is [0, *MAXPAY*], where the parameter *MAXPAY* is decided by the team leader. *MAXPAY* denotes the maximum payment that an agent team can afford to keep a single agent as a team member.

Membership Functions for Input Parameters:

For Utilisation Ratio, the following four linguistic states [17] are selected and expressed by appropriate fuzzy sets: *Never* (N), *Seldom* (S), *Medium*, (M) and *Frequent* (F). Another input parameter Contribution Ratio also has four linguistic states, these being *None* (N), *Little* (L), *Medium* (M) and *Huge* (H). The trapezoidal [17] fuzzy membership function is adopted here to define fuzzy memberships of these

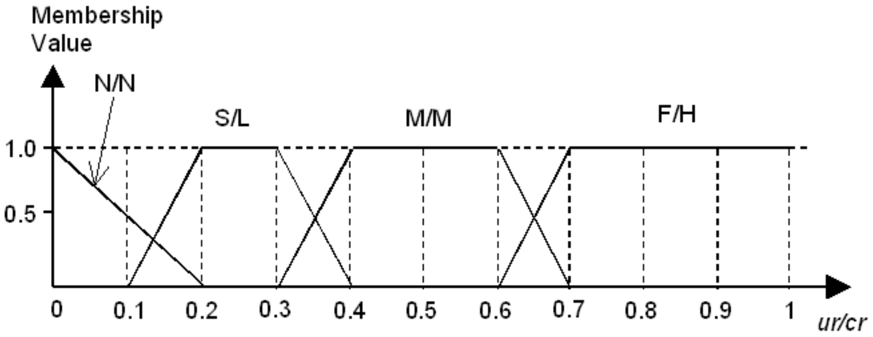


Fig. 2 Fuzzy Membership Function for *ur/cr*

four fuzzy sets. The membership functions for these four fuzzy sets are defined in Equations 4 through 7, respectively. They are also depicted in Figure 2.

$$F_{Never}(x)/F_{None}(x) = \begin{cases} 1-5x & x \in [0,0.2] \\ 0 & x \notin [0,0.2] \end{cases} \tag{4}$$

$$F_{Seldom}(x)/F_{Little}(x) = \begin{cases} 10x-1 & x \in [0.1,0.2] \\ 1 & x \in (0.2,0.3) \\ 4-10x & x \in [0.3,0.4] \\ 0 & x \notin [0.1,0.4] \end{cases} \tag{5}$$

$$F_{Medium}(x) = \begin{cases} 10x-3 & x \in [0.3,0.4] \\ 1 & x \in (0.4,0.6) \\ 7-10x & x \in [0.6,0.7] \\ 0 & x \notin [0.3,0.7] \end{cases} \tag{6}$$

$$F_{Frequent}(x)/F_{Huge}(x) = \begin{cases} 10x-6 & x \in [0.6,0.7] \\ 1 & x \in (0.7,1] \\ 0 & x \notin [0.6,1] \end{cases} \tag{7}$$

For *ara*, three linguistic states are selected, namely *Rare* (R), *Some* (S), and *Many* (M). The membership functions for *ara* are defined in Equations 8 through 10, and depicted in Figure 3.

$$F_{Rare}(x) = \begin{cases} 1-4x & x \in [0,0.4] \\ 0 & x \notin [0,0.4] \end{cases} \tag{8}$$

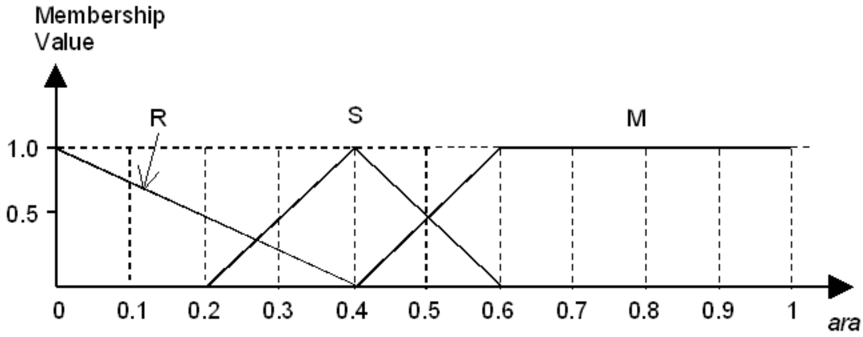


Fig. 3 Fuzzy Membership Function for *ara*

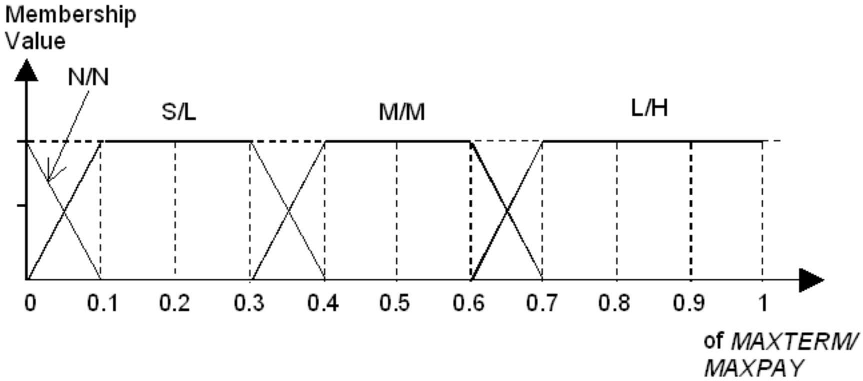


Fig. 4 Fuzzy Membership Function for *ct/cl*

$$F_{Some}(x) = \begin{cases} 5x-1 & x \in [0.2, 0.4] \\ 3-5x & x \in (0.4, 0.6] \\ 0 & x \notin [0.2, 0.6] \end{cases} \tag{9}$$

$$F_{Many}(x) = \begin{cases} 5x-2 & x \in [0.4, 0.6] \\ 1 & x \in (0.6, 1] \\ 0 & x \notin [0.4, 1] \end{cases} \tag{10}$$

Membership Functions for Output Parameters:

There are two output parameters – *Contract Term* (*ct*) and *Commission Level* (*cl*) – in the fuzzy method. For *ct*, the following four linguistic states are selected: *Long* (L), *Medium* (M), *Short* (S) and *No* (N). For *cl*, *High* (H), *Medium* (M), *Low* (L) and *No* (N) are chosen as the four linguistic states. Fuzzy membership functions of these fuzzy sets are defined in Equations 11 through 14, and described in Figure 4.

$$F_{No}(x) = \begin{cases} 1 - 10x & x \in [0, 0.1] \\ 0 & x \notin [0, 0.1] \end{cases} \tag{11}$$

$$F_{Short}(x)/F_{Low}(x) = \begin{cases} 10x & x \in [0, 0.1] \\ 1 & x \in (0.1, 0.3) \\ 4 - 10x & x \in [0.3, 0.4] \\ 0 & x \notin [0, 0.4] \end{cases} \tag{12}$$

$$F_{Medium}(x) = \begin{cases} 10x - 3 & x \in [0.3, 0.4] \\ 1 & x \in (0.4, 0.6) \\ 4 - 10x & x \in [0.6, 0.7] \\ 0 & x \notin [0.3, 0.7] \end{cases} \tag{13}$$

$$F_{Long}(x)/F_{High}(x) = \begin{cases} 10x - 6 & x \in [0.6, 0.7] \\ 1 & x \in (0.7, 1] \\ 0 & x \notin [0.6, 1] \end{cases} \tag{14}$$

Fuzzy Rule Base:

A fuzzy rule base is a matrix of combinations of each of the input linguistic parameters and their corresponding output parameters. The rule base in this mechanism is described in Table 3.

Table 3 Fuzzy Rule Base Matrix

Agent Resource Availability		***R***	***S***	***M***
Utilisation Ratio	***Contribution Ratio***	**Output Parameters:** ***ct, cl***		
N	***N***	*ct=N, cl=N*	*ct=N, cl=N*	*ct=N, cl=N*
N	***L***	*ct=M, cl=L*	*ct=N, cl=N*	*ct=N, cl=N*
N	***M***	n/a	n/a	n/a
N	***H***	n/a	n/a	n/a
S	***N***	*ct=M, cl=L*	*ct=N, cl=N*	*ct=N, cl=N*
S	***L***	*ct=L, cl=L*	*ct=S, cl=L*	*ct=N, cl=N*
S	***M***	*ct=L, cl=L*	*ct=M, cl=M*	*ct=S, cl=M*
S	***H***	*ct=L, cl=M*	*ct=S, cl=M*	*ct=N, cl=M*
M	***N***	n/a	n/a	n/a
M	***L***	*ct=L, cl=M*	*ct=M, cl=L*	*ct=S, cl=L*
M	***M***	*ct=L, cl=M*	*ct=M, cl=M*	*ct=M, cl=L*
M	***H***	*ct=L, cl=H*	*ct=L, cl=M*	*ct=M, cl=M*
F	***N***	n/a	n/a	n/a
F	***L***	*ct=L, cl=M*	*ct=M, cl=M*	*ct=L, cl=L*
F	***M***	*ct=L, cl=H*	*ct=L, cl=M*	*ct=L, cl=L*
F	***H***	*ct=L, cl=H*	*ct=L, cl=H*	*ct=L, cl=M*

Determination of Output Membership Values and Defuzzification

Each entry of the rule base is a rule, which is defined by ANDing two linguistic input parameters to produce an output combination, in the form of: $IF(F(ur) = \alpha$ AND $F(cr) = \beta$ AND $F(ara) = \gamma)$ THEN $(F(ct) = \delta)$ AND $F(cl) = eta)$, where $\alpha \in \{$*Never, Seldom, Medium, Frequent*$\}$, $\beta \in \{$*None, Little, Medium, Large*$\}$, $\gamma \in \{$*Rare, Some, Many*$\}$, $\delta \in \{$*Long, Medium, Short, No*$\}$, and $\eta \in \{$*High, Medium, Low, No*$\}$. In this mechanism, the *AND/MIN* operator is used to combine the membership values, i.e. the weakest membership determines the degree of membership in the intersection of fuzzy sets [8] [17]. Hence, the output membership value $\mu_{\delta/\eta}(v)$ can be calculated using Equation 15.

$$\mu_{\delta/\eta}(v) = MIN(\mu_\alpha(ur), \mu_\beta(cr), \mu_\gamma(ara)) \tag{15}$$

With regard to output membership, the output values can be determined by tracing the membership values for each rule back through the output membership functions. Finally, the *centroid* defuzzification method [8] [17] is used to determine the output value. In *centroid* defuzzification, the output value is calculated using Equation 16, where membership of v_i is represented as $\mu(v_i)$, and k is the number of fuzzy rules which are activated.

$$DF = \frac{\sum_{i=1}^{k}(v_i \cdot \mu(v_i))}{\sum_{i=1}^{k}\mu(v_i)} \tag{16}$$

3.4 Experiments

To analyse the performance of the flexible team-formation mechanism, some experiments are conducted to compare it with the one-shot and long-term team-formation mechanisms. The experimental environment is set up to simulate the scenario introduced in Subsection 3.1. Each agent possesses one (or more) kind of resource(s), and needs to contribute its resource(s) to achieve rewards through accomplishing tasks in the system. However in most cases an agent cannot accomplish a task due to its limited resource(s). Hence, agents need to cooperate with others in order to realise their goals. This experiment simulates some real world applications. For example, in a Web service system [24], each peer can only provide a limited number of services (i.e. possesses limited resources). To execute a complex task, we need to aggregate or combine small services in different peers into larger services (i.e. form a team to perform the task).

In the experiment, a set of tasks is sent to the agents, and they perform these tasks using one-shot, long-term and flexible team-formation mechanisms, respectively. In order to avoid agent teams including too many agents for too long a time (especially for long-term teams), we set a maximum team size to limit the number of long-term team members. In this experiment, the maximum team size equals five, which means an agent team can at most keep five long-term members. Two factors are compared in the experiment, namely *Agent Searching Time* and *Reward Distribution Situation*.

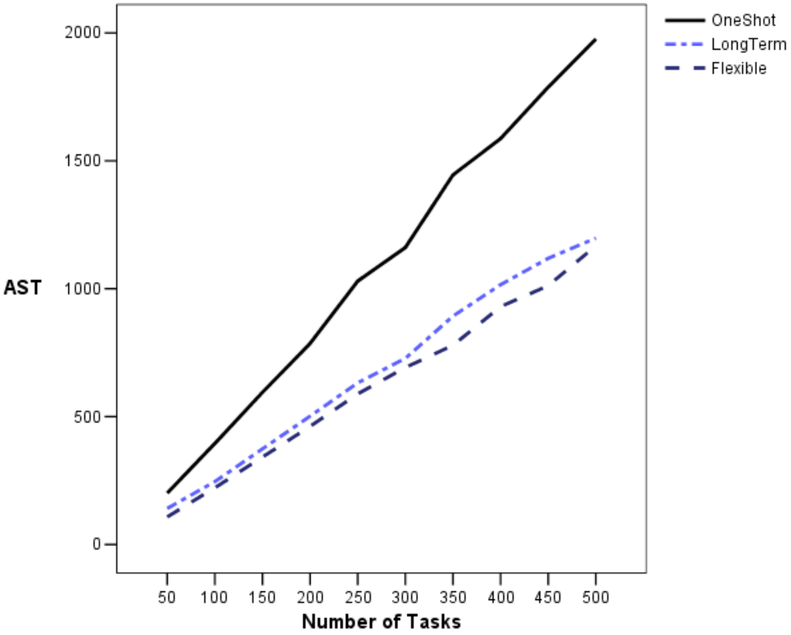


Fig. 5 Agent Search Comparison (no. of searches vs. no. of tasks)

Agent Searching Time represents the time that a team leader needs to search for required agents from the agent board to accomplish the tasks. In general, the higher the *Agent Searching Time*, the more communication cost the team leader needs to spend on searching agents.

According to the experimental result, it can be seen that the *Agent Searching Time* of one-shot team formation is much higher than both long-term and flexible team formation (See Figure 5). This is because team leaders in one-shot teams need to keep searching suitable team members for each task and disband them after a task is accomplished. With long-term and flexible team formation, the whole team (or part thereof) is retained after a task is completed. Thus these two latter strategies will have less communication overhead. The experimental result shows that long-term teams have higher *Agent Searching Time* than flexible teams. This is because, in the experiment, a long-term team can at most keep a limited number of members for a long period. Hence, after a team accomplishes several tasks, the number of long-term members will reach the limit, and the team will start to search and disband new members in subsequent tasks. The result shows that the *Agent Searching Time* of using flexible team formation is the lowest, which means it has the lowest communication overhead among the three mechanisms.

Reward Distribution Situation is the second comparison factor. It represents the rationality of agent team organisation. Without considering communication overhead, a one-shot team has an ideal organisational structure because all its team

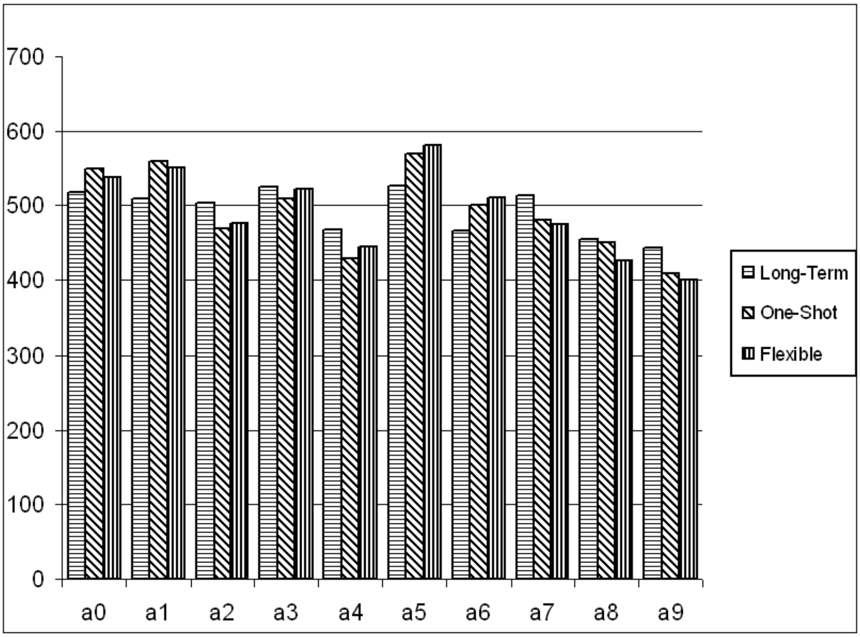


Fig. 6 Reward Distribution Situation (no. of reward units per agent)

members contribute to task executions. Hence the *Reward Distribution Situation* of one-shot teams can be considered as the benchmark for team organisation rationality. Throughout this experiment, it can be seen that the *Reward Distribution Situation* of flexible teams is closer to one-shot teams than long-term teams (See Figure 6). Therefore, flexible teams have more reasonable organisations than long-term teams.

From the experimental results, it can be seen that the flexible team-formation mechanism is more suitable for self-interested agents and open environments. In cooperative domains, agents do not care whether the reward is distributed rationally, the most important thing that cooperative agents consider is the overall benefit to the team. However, self-interested agents do consider rationality of reward distribution, and do not want to keep "less valuable" members in the team for a long period. The flexible mechanism can enable agent teams to keep valuable team members according to their performance and changing environments. Furthermore, agent teams can adjust their long-term member selection criteria through modifying the member evaluation parameters. This feature can make team formation more flexible and suitable for open environments. Therefore, compared with one-shot and long-term team-formation mechanisms, the flexible team-formation mechanism can enable self-interested agents to form more reasonable teams in an open environment with less communication overhead.

3.5 Summary

As a social entity, an intelligent agent needs to cooperate with others in most multi-agent environments. At the same time, unreasonable team-formation mechanisms could prevent agents from achieving local benefits, or lead to unnecessary system overhead. Focusing on challenges inherent in dynamic application domains, many researchers have suggested using long-term or one-shot team-formation mechanisms in MASs. However, both of these mechanisms have some advantages and disadvantages, as discussed earlier. A flexible team-formation mechanism can avoid some of the limitations of the one-shot and long-term team-formation mechanisms. It can enable agents to automatically evaluate the performance of other agents in the system, and select team members with reasonable terms and costs according to the evaluation result. In flexible team-formation, factors related to agent performance and task requirements are considered as evaluation factors. Through evaluating these factors, team compositions are more reasonable and can avoid some potential benefit conflicts among team members.

4 Collaborative Problem Solving through Agent Competition

In some application domains, agent competition can also be involved in collaborative problems. Suppose a set of autonomous agents has a global goal it wants to achieve, where this goal is too complex to be achieved by any single agent. Therefore, the global goal must be divided into several local goals and distributed to agents by considering their individual ability, requirement, restriction etc. Now each agent wants to minimize its costs, that is, prefers to do as little as possible. Therefore, even though the agents have a common goal, there is actually a conflict of interest here. Agents may argue and compete with each other in order to maximize their individual benefits and also ensure that the global goal be achieved in a timely manner. This kind of competition within a collaborative problem may pertain in applications such as resource allocation, task distribution, emergencies etc. Agent negotiation can be employed to solve competition problems.

4.1 Traditional Agent Negotiation

Motivations and aims determine agent behavior in negotiation. Therefore, it is necessary to discuss the kinds of agent behavior which can take place during negotiations. In general, agents may compete or cooperate with each other in order to reach their own goals or a common goal within a MAS. Final agreements about how to compete or cooperate are achieved through negotiation. Therefore, negotiations can be classified into *competitive* and *cooperative* according to the behaviors of its participants. In a *competitive* negotiation, participants perform the role of challengers, while in a *cooperative* negotiation, participants act as cooperators. However, both kinds of negotiation contain the following four components in general [33] :

1. The negotiation protocol,
2. The negotiation strategies,

3. The information state of agents,
4. The negotiation equilibrium.

The negotiation protocol specifies the rules of engagement in agent negotiation. It defines what kinds of (i) interaction between agents can be taken in different circumstances; (ii) sequences are allowed and (iii) deals can be made in the negotiation. For example, Rubinstein's alternating offers protocol is a very commonly used negotiation protocol. In this protocol, one of the negotiation participants makes an offer, then the other responds by either accepting the offer, rejecting it, or opting out of the negotiation. The negotiation will be finished only when all negotiation participants accept an offer, or one or more negotiation participants opt out. In general, agents should make an agreement on the negotiation protocol before the negotiation proper starts. The negotiation protocol will be designed differently by considering the following factors: (a) numbers of negotiation participants (e.g. sellers and buyers), (b) numbers of negotiation issues (e.g. a car's price, color, model and etc.), and (c) negotiation environment (buyers' market or sellers' market).

The negotiation strategy specifies the sequence of actions that the negotiation participants plan to make during the negotiation. In competition problem negotiation, agents try to maximize their own local interests during the negotiation, and also have to ensure the global goal of the negotiation. Therefore, agents may employ different negotiation strategies by considering self and/or other information. For example, an agent could bargain very hard throughout the negotiation in order to maximize its benefit or give some kind of concession under time restrictions. Also, it should be clear that a strategy which performs well with certain protocols may not necessarily do so with others. Therefore, both the negotiation scenario and protocol in use should be considered when the negotiation participant chooses a negotiation strategy.

The agents' information state describes information about the negotiation, which can be classified as 'private' and 'public' [9]. Private information describes an agent's self situation, such as the negotiation strategy, which is only possessed by that particular agent. Unless the negotiation participant agrees to share its private information with others, it is not reachable by other negotiation participants. Public information describes the negotiation environment, such as the number of negotiation participants, number of negotiation issues, negotiation protocols etc. This public information is available to all negotiation participants. In the negotiation, if all negotiation participants would like to share all their private information, then the negotiation is referred to as 'negotiation with complete information'. Otherwise, if the negotiation participant does not want to share their private information, then the negotiation is termed 'negotiation with incomplete information'. An agent's information state will impact the agent's choice of negotiation strategy.

When agents choose negotiation protocols and negotiation strategies, agents create negotiation mechanisms. During the negotiation, the negotiation mechanism must be stable, i.e. a strategy profile must constitute an equilibrium. The Nash equilibrium [26] is a commonly used concept. Two strategies are in Nash equilibrium if each negotiation participant's strategy is the best response to its opponent's strategy. The equilibrium is a very important and necessary condition for negotiation system stability. For different negotiation protocols, the equilibrium strategy may differ. However, it is required that each negotiation participant should select an equilibrium strategy in the negotiation.

In this subsection, we provide an example of negotiation between two agents. In our example, the negotiation is performed between two agents, i.e. the 'buyer' agent and the 'seller' agent. Both agents are bargaining over the price, therefore it is a single-issue negotiation. In the following, we will show the four components in our example negotiation and introduce how the negotiation is processed.

The negotiation protocol. We simply adopt the basic alternating offers protocol [28]. Let b denote the buyer agent, and s the seller agent. The negotiation starts when the first offer is made by an agent (b or s). The agent who makes the initial offer is selected randomly at the beginning of the negotiation. When an agent receives an offer from its opponent, it will evaluate it. According to this evaluation, the agent will take one of the following actions: (i) *Accept*: when the value of the offer received from the opponent is equal to or greater than the value of the counter-offer it is going to send in the next negotiation cycle. Once the agent accepts this offer, the negotiation ends successfully in an agreement; (ii) *Reject*: when the value of the offer received from the opponent is less than the value of the counter-offer it is going to send in the next negotiation cycle. Once the agent rejects this offer, providing the negotiation deadline has not been reached, the agent sends out a counter-offer to its opponent and the negotiation proceeds to the next cycle; (iii) *Quit*: when the negotiation deadline falls due and no agreement has been reached, then the agent has to quit and the negotiation fails.

The negotiation strategies. In our example, two agents are bargaining over price, therefore each agent should have some idea about its acceptability. Let $[IP^a, RP^a]$ denote the range of price values which are acceptable to agent a, where $a \in \{b, s\}$. IP^a denotes the initial price and RP^a the reserve price of agent a. In general, when $a = b$, $IP^b \leq RP^b$, and when $a = s$ $IP^b \geq RP^b$. Let $\hat{a}$ denote agent a's opponent, where $\hat{a} \in \{b, s\}$. Then the offer made by agent a to agent $\hat{a}$ at time t ($0 \leq t \leq \tau^a$), where τ^a is the deadline for agent a, is modeled as a function Φ^a depending on time as follows:

$$p^t_{a \to \hat{a}} = \begin{cases} IP^a + \Phi^a(t)(RP^a - IP^a) & a = b \\ RP^a + (1 - \Phi^a(t))(IP^a - RP^a) & a = s \end{cases} \tag{17}$$

where function $\Phi^a(t)$ ($0 \leq \Phi^a(t) \leq 1$) is called the negotiation decision function (NDF) [12]. The common way to define $\Phi^a(t)$ is:

$$\Phi^a(t) = k^a + (1 - k^a)(\frac{t}{\tau^a})^{1/\lambda} \tag{18}$$

where k^a ($0 \leq k^a \leq 1$) is the parameter which controls the initial offer. For example, when $k^a = 0$, the initial offer is IP^a, and when $k^a = 1$, the initial offer is RP^a; λ is the parameter which controls the agent behavior. Depending on the value of λ, three extreme cases show different patterns of behavior for the agent

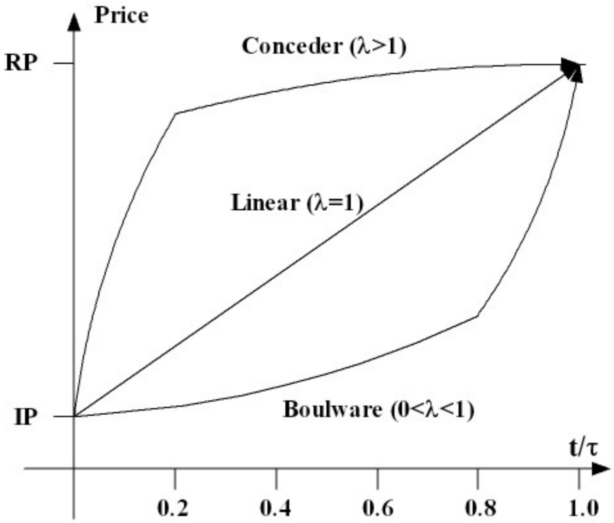


Fig. 7 Negotiation decision function for the buyer

b in Figure 7 [9]: (i) *Conceder*: when $\lambda > 1$, agent b gives more concession in the beginning of the negotiation, and less concession closer to the deadline; (ii) *Linear*: when $\lambda = 1$, agent b gives constant concession throughout the negotiation; and (iii) *Boulware*: when $0 \leq \lambda \leq 1$, agent b gives less concession initially, and more concession when the deadline is looming.

Finally, agent utility functions at time t are defined as per Equation 19.

$$U^a(p^t_{a\rightarrow\hat{a}}) = \begin{cases} RP^a - p^t_{a\rightarrow\hat{a}} & a = b \\ p^t_{a\rightarrow\hat{a}} - RP^a & a = s \end{cases} \tag{19}$$

$U^a(t)$ is the agent a's evaluation result of its opponent's offer at negotiation cycle t; based on this evaluation result, agent a can make a decision about its action.

The information state of agents. The sample negotiation is a negotiation with incomplete information, i.e. both agents s and b do not share their private information with each other.

The negotiation equilibrium. The Nash equilibrium is employed in our sample negotiation. The action, A^a, of agent a at time t is defined as follows:

$$A^a(p^t_{a\rightarrow\hat{a}}) = \begin{cases} Quit & if t > \tau^a, \\ Accept & if U^a(p^t_{\hat{a}\rightarrow a}) \geq U^a(p^{t'}_{a\rightarrow\hat{a}}), \\ Reject & if U^a(p^t_{\hat{a}\rightarrow a}) < U^a(p^{t'}_{a\rightarrow\hat{a}}). \end{cases} \tag{20}$$

where t' is the time of the next negotiation cycle. Therefore, the equilibrium strategy employed in this negotiation indicates that the agent will only accept the offer which can maximize self's benefit given the time constraint.

4.2 Partner Selection in Agent Negotiation

In the previous subsection, we briefly introduced agent negotiation and also indicated that it can be employed by agents to solve competition problems. However, due to the rapid development of autonomous agents and Internet techniques, most MAS work environments have become uncertain and dynamic. In such open and dynamic environments, when the number of potential partners is huge, performing complicated traditional negotiations with all potential partners may be expensive in terms of computational time and resources – indeed even impractical. Thus, we introduce an approach which can be employed by agents to choose partners from a large pool of potential partners with a high chance of reaching a good agreement in subsequent negotiations.

Agents may have different criteria on partner selection based on the purpose of their negotiation. Generally, in *cooperative* negotiation, agents will select a partner which will increase global benefits; while in *competitive* negotiation, agents prefer some partners which can supply the highest benefit to themselves. However, researchers have found that it is not always beneficial for agents to only cooperate with others about global tasks in *cooperative* negotiation [16] [42]. Also, in a *competitive* negotiation, agents should consider the global tasks. Furthermore, when agent behaviors are in between these two extreme cases, the existing partners selection approach is no longer suitable.

Zhang et al. proposed a dual concern model which provides an outline about the degrees of concern of an agent for its own and other's outcomes [43]. However, this model only briefly mentions the main trend of these degrees, without offering any calculation or comparison method. To address these problems, we further extended this dual concern model to allow agents to make reasonable decisions on their behaviors during partner selection based on these degrees. The extended dual concern model is shown in Figure 8.

In Figure 8, the x-axis indicates the percentage of self-concern of an agent while the y-axis is the percentage of other-concern from the agent. θ represents a *ReliantDegree* (i.e. reflection of the collaboration degree), where $\theta \in [0^\circ, 90^\circ]$. We use *selfishness* to represent the percentage of self-concern of an agent, which can be calculated by $\cos(\theta)$, and *selflessness* to represent the percentage of other-concern, which can be evaluated by $\sin(\theta)$. A *ReliantDegree* can illustrate the level of collaboration between the agent and its potential partner. From the extended model, we find that there are two extreme cases: (i) when the agent only emphasizes its own outcome, its negotiation attitude is completely *selfish* ($\theta = 0^\circ$); and (ii) when the agent only cares about its partner's outcome, its attitude is completely *selfless* ($\theta = 90^\circ$). From this model, it is clear that there are many other cases between completely *selfish* and completely *selfless* behaviors.

Suppose that there are n potential partners for an agent ID_x in a MAS. If we use a four-tuple p_i^x to represent the ith potential partner of agent ID_x, p_i^x can be formally defined by Equation 21:

$$p_i^x = < ID_i, GainRatio_i^x, ContributionRatio_i^x, ReliantDegree_i^x > \tag{21}$$

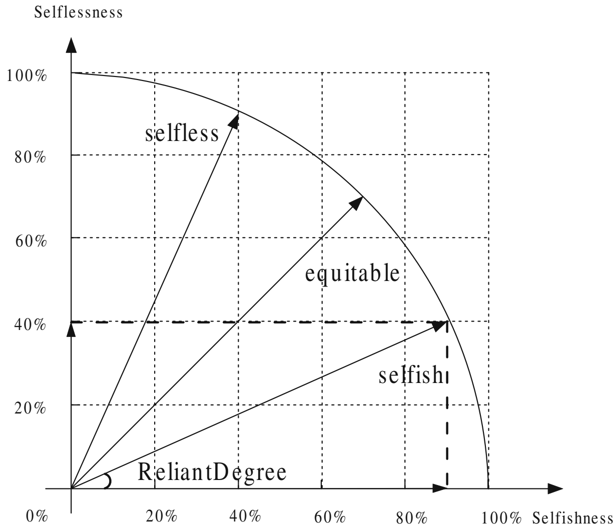


Fig. 8 The extended dual concern model

where ID_i is the unique identification of the ith potential partner, and $GainRatio_i^x$, $ContributionRatio_i^x$ and $ReliantDegree_i^x$ are factors used to evaluate the potential partner ID_i to be selected in the negotiation. These three factors are defined in Definitions 14 through 16, respectively.

Definition 14. $GainRatio_i^x$ is the percentage benefit that agent ID_x obtains out of the global benefit upon completion of the task. $GainRatio_i^x$ can be calculated as $GainRatio_i^x = \frac{S}{L} \times 100\%$, $GainRatio_i^x \in [0, 100\%]$, where S denotes the benefit that agent ID_x gains by selecting partner agent ID_i as its partner, and L denotes the global benefit by completing the task.

Definition 15. $ContributionRatio_i^x$ is the percentage benefit that agent ID_i obtains out of the global benefit upon completion of the task. $ContributionRatio_i^x$ can be calculated as $ContributionRatio_i^x = \frac{I}{L} \times 100\%$, $ContributionRatio_i^x \in [0, 100\%]$, where I denotes the benefit that partner agent ID_i gains by cooperating with agent ID_x, and L denotes the global benefit by completing the task.

Definition 16. $ReliantDegree_i^x$ represents agent ID_x's attitude to the negotiation, and also indicates the dynamic behavior of the agent, such as selfishness, selflessness, or other. $ReliantDegree_i^x$ can be calculated as $ReliantDegree_i^x = \arctan(\frac{Cr_x^i}{Cr_i^x})$, $ReliantDegree \in [0°, 90°]$, where Cr_x^i indicates how much agent ID_x trusts partner agent ID_i, which can be defined as the trading success ratio from partner agent ID_x to ID_i, or can be assigned by the system based on the performance record of partner agent ID_i, and Cr_x^i indicates how much partner agent ID_i trusts agent ID_x, which can be defined in the similar way as Cr_x^i.

Then an agent ID_x's evaluation of its potential partner ID_i is represented by $CollaborateDegree_i^x$, which is defined as follows:

$$CollaborateDegree_i^x = \Psi(p_i^x) \tag{22}$$

where $CollaborateDegree_i^x \in [0,1]$. This indicates the tendency that agent ID_i will be selected as a partner in subsequent negotiations by agent ID_x. The bigger the $CollaborateDegree_i^x$, the higher the likelihood that agent ID_i will be selected. The function Ψ specifies how to evaluate a potential partner. The interested reader can refer to [31] for a (non-linear) fuzzy approach to Ψ. In this chapter, we only consider a linear approach to Ψ.

In order to cover all potential cases in partner selection, we need to consider not only both *GainRatio* and *ContributionRatio*, but also the preference of the agent on these two criteria. It is proposed that the agent's preference on *GainRatio* and *ContributionRatio* can be represented by a normalized weight. Let w_g stand for the weight on *GainRatio*, w_c stand for the weight on *ContributionRatio*, and $w_c + w_g = 1$. Then the *CollaborationDegree* between agent ID_x and its potential partner ID_i is defined as follows:

$$CollaborateDegree_i^x \;=\; GainRatio_i^x \times w_g + ContributionRatio_i^x \times w_c \tag{23}$$

The *collaborationDegree* ($\in [0,1]$) indicates the degree for which the potential partner should be selected by the agent. The bigger the *collaborationDegree*, the more chance that the potential partner will be selected by the agent. In general, there are three extreme cases on different combinations of w_c and w_g, namely:

- When $w_g = 0$ and $w_c = 1$, *CollaborateDegree* is calculated based only on *ContributionRatio*, i. e. agent ID_x's attitude to negotiation is fully *selfless*.
- When $w_g = 1$ and $w_c = 0$, *CollaborateDegree* is calculated based only on *GainRatio*, i. e. agent ID_x's attitude to negotiation is fully *selfish*.
- When $w_g = w_c = 0.5$, *CollaborateDegree* is calculated based equally on *GainRatio* and *ContributionRatio*, i. e. agent ID_x's attitude to negotiation is *equitable*.
- Besides the above three cases, the restriction of $w_g + w_c = 1$ can also reflect agent ID_x's attitude to *GainRatio* and *ContributionRatio* in other cases.

The weights w_g and w_c can be calculated by employing the value of *ReliantDegree*, which are defined by Equation 24 and Equation 25, respectively.

$$w_g = \cos^2(ReliantDegree) \tag{24}$$

$$w_c = \sin^2(ReliantDegree) \tag{25}$$

Finally, by combining Equations 23 through 25, the potential partners are evaluated by considering the factors of *GainRatio*, *ContributionRatio* and *ReliantDegree*. The *collaborationDegree* between the agent ID_x and its potential partner ID_i is:

$$CollaborateDegree_i^x = GainRatio_i^x \times \cos^2(ReliantDegree_i^x) + ContributionRatio_i^x \times \sin^2(ReliantDegree_i^x) \quad (26)$$

where $CollaborateDegree_i^x \in [0,1]$. Then the collaboration degrees set (*CollaborateDegree*^{x}) between the agent ID_x and all its potential partners are generated according to Equation 27.

$$CollaborateDegree^x = \{CollaborateDegree_i^x\}, i \in [1,n] \quad (27)$$

Finally, any sorting algorithm can be employed to select favorable partners or exclude unsuitable partners from the collaboration degree set $CollaborateDegree^x$ for the agent ID_x.

In this chapter, three examples are demonstrated. In each example, agent g is going to select the most suitable partner from three potential partners (agents g_a, g_b and g_c). These examples will illustrate how the proposed approach selects the most suitable partner for the agent.

Table 4 Example 1

Agent	*Gain Ratio*	*Contribution Ratio*	*Reliant Degree*	*Collaborate Degree*
g_a	80%	20%	0°	0.8
g_b	50%	50%	0°	0.5
g_c	20%	80%	0°	0.2

In Example 1 (Table 4), as the agent ID_x performs as a fully *selfish* agent ($w_g = \cos^2(0°) = 1$ and $w_c = \sin^2(0°) = 0$), the potential partner who can offer the biggest *GainRatio* will be selected as the most suitable partner. From Table 4, agent g_a should be selected as the most suitable partner because it can contribute the highest *GainRatio* to agent ID_x among the three potential partners. By using our proposed Equation 26, agent g_a is also chosen as the most suitable partner because the *CollaborateDegree* for agent g_a is the largest among the three potential partners.

In Example 2 (Table 5), as the agent ID_x performs as a fully *selfless* agent ($w_g = \cos^2(90°) = 0$ and $w_c = \sin^2(90°) = 1$), agent g_c should be selected as the most suitable partner because it has the largest *ContributionRatio*. According to Equation 26, agent g_c is also selected as the most suitable partner because the *CollaborateDegree* for agent g_c is the largest among the three potential partners.

Table 5 Example 2

Agent	*Gain Ratio*	*Contribution Ratio*	*Reliant Degree*	*Collaborate Degree*
g_a	80%	20%	90°	0.2
g_b	50%	50%	90°	0.5
g_c	20%	80%	90°	0.8

Table 6 Example 3

Agent	*Gain Ratio*	*Contribution Ratio*	*Reliant Degree*	*Collaborate Degree*
g_a	80%	20%	0°	0.8
g_b	80%	20%	45°	0.5
g_c	80%	20%	90°	0.2

In Example 3 (Table 6), the agent ID_x has different attitudes to its potential partners. For potential partner g_a, agent ID_x performs as a *selfish* agent so that only the *GainRatio* (80%) will be used to select the most suitable partner. For potential partner g_b, agent ID_x performs as an *equitable* agent so that both *GainRatio* (80%) and *ContributionRatio* (20%) will be used to evaluate whether g_b could be chosen as a suitable partner. Therefore, the final benefit by considering both *GainRatio* and *ContributionRatio* for g_b should be between 20% and 80%. For potential partner g_c, agent ID_x performs as a *selfless* agent so that only the benefit of *ContributionRatio* (20%) will be used for the selection of g_c as a partner. By comparing the three cases, agent g_a should be selected as the most suitable partner because agent ID_x would gain the largest benefit(80%) when collaborating with agent g_a. According to Equation 26, agent g_a is also selected as the most suitable partner because the *CollaborateDegree* for agent g_a is the largest among the three potential partners.

Therefore, from the examples, it can be seen that by considering the factors of *GainRatio*, *ContributionRatio* and *ReliantDegree* between the agent and its potential partners, a partner selection mechanism can be generated dynamically to allow agents to adapt to their individual behaviors in negotiation. The selection result is also accurate and reasonable.

4.3 Behavior Prediction in Agent Negotiation

Negotiation is a means for agents to communicate and compromise to reach mutually beneficial agreements [10] [19]. However, in most situations, agents do not possess complete information about their partners' negotiation strategies, and may have difficulty in making a decision on future negotiation, such as how to select suitable partners [3] [25], or how to generate a suitable offer in the next negotiation cycle [29]. Therefore estimation approaches which can predict uncertain situations and possible changes in the future are required to help agents to generate good and efficient negotiation strategies. Research on partners' behavior estimation has been a very active area in recent years. Several estimation strategies have been proposed [6] [7] [41]. However, as these estimation strategies are used in real applications, some limitations begin to emerge, such as inaccurately estimated results or substantial time cost.

Machine Learning is a popular mechanism adopted by researchers in agent behavior estimation. In general, this kind of approach comprises two steps in order to estimate an agents' behavior. In the first step, the proposed estimation function is required to be well trained using the available training data. Therefore, in a way, the performance of the estimation function is virtually determined by the training result. In this step, as much data as possible is employed by designers to train a system. The training data could be synthetic and/or collected from the real world. Usually, synthetic data is helpful in training a function to enhance its problem solving ability for some particular issues, while real world data can help the function to improve its ability in complex problem solving. After the system has been trained, the second step is to employ the estimation function to predict partner behavior in the future. However, no matter which and how many data are employed by designers to train the proposed function, the training data will never be sufficiently comprehensive to cover all situations in reality. Therefore, even though an estimation function is well trained, it is also quite possible that some estimation results do not make sense at all for some kind of agents whose behavior records are not included in the training data. Currently, as negotiation environments become more open and dynamic, agents with different kinds of purpose, preference and negotiation strategy can enter and leave the negotiation dynamically. This Machine Learning-based agent behavior estimation function may not work well in some more flexible application domains, for reasons of (i) lack of sufficient data to train the system, and (ii) requiring too many resources during each training process.

In order to address the aforementioned issues, in this subsection we introduce a quadratic regression approach for analysis and estimation of partner behaviors during negotiation. The proposed quadratic regression function is:

$$u = a \times t^2 + b \times t + c \tag{28}$$

where u is the expected utility gained from a partner, t $(0 \leq t \leq \tau)$ is the negotiation cycle and a, b and c are parameters which are independent of t. It is noticed that the three types of agents' behaviors in Figure 7 can be represented by this function as follows:

- $a > 0$ (Boulware): the rate of change in the slope is increasing, corresponding to smaller concession in the early cycles but large concession in later cycles.
- $a = 0$ and $b \neq 0$ (Linear): the rate of change in the slope is zero, corresponding to making constant concession throughout the negotiation.
- $a < 0$ (Conceder): the rate of change in the slope is decreasing, corresponding to large concession in early cycles, but smaller concession in later cycles.

We firstly transfer the proposed quadratic function 28 to a linear function, as follows. Let

$$\begin{cases} x = t^2 \\ y = t \end{cases} \tag{29}$$

Then Equation 29 can be rewritten as:

$$u = a \times x + b \times y + c \tag{30}$$

where both a and b are independent of variables x and y. Let pairs $(t_0,\hat{u_0})$, …, $(t_n,\hat{u_n})$ be instances from each negotiation cycle. The distance (ε) between the real utility value ($\hat{u_i}$) and the expected value (u_i) should obey the Gaussian distribution $\varepsilon \sim N(0,\sigma^2)$, where $\varepsilon = \hat{u_i} - a \times x_i - b \times y_i - c$. Now since each $\hat{u_i} = a \times x_i + b \times y_i + c + \varepsilon_i$, $\varepsilon_i \sim N(0,\sigma^2)$, $\hat{u_i}$ is distinctive, and the joint probability density function for $\hat{u_i}$ is:

$$L = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp[-\frac{1}{2\sigma^2}(\hat{u_i} - ax_i - by_i - c)^2] \tag{31}$$
$$= (\frac{1}{\sigma\sqrt{2\pi}})^n \exp[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(\hat{u_i} - ax_i - by_i - c)^2]$$

where L indicates the probability that a particular $\hat{u_i}$ may occur. Because each $\hat{u_i}$ comes from the historical record, we should use their probabilities as L's maximum value. Obviously, in order to make L achieve its maximum, $\sum_{i=1}^{n}(\hat{u_i} - ax_i - by_i - c)^2$ should achieve its minimum value. Let

$$Q(a,b,c) = \sum_{i=1}^{n}(\hat{u_i} - ax_i - by_i - c)^2 \tag{32}$$

We calculate the first-order partial derivative for $Q(a,b,c)$ on a, b and c respectively, and let their results equal zero, as follows:

$$\begin{cases} \frac{\partial Q}{\partial a} = -2\sum_{i=1}^{n}(\hat{u_i} - ax_i - by_i - c)x_i = 0 \\ \frac{\partial Q}{\partial b} = -2\sum_{i=1}^{n}(\hat{u_i} - ax_i - by_i - c)y_i = 0 \\ \frac{\partial Q}{\partial c} = -2\sum_{i=1}^{n}(\hat{u_i} - ax_i - by_i - c) = 0 \end{cases} \tag{33}$$

Then the Equations 33 can be expanded to:

$$\begin{cases} (\sum_{i=1}^{n} x_i^2)a + (\sum_{i=1}^{n} x_i y_i)b + (\sum_{i=1}^{n} x_i)c = \sum_{i=1}^{n} x_i \hat{u_i} \\ (\sum_{i=1}^{n} x_i y_i)a + (\sum_{i=1}^{n} y_i^2)b + (\sum_{i=1}^{n} y_i)c = \sum_{i=1}^{n} y_i \hat{u_i} \\ (\sum_{i=1}^{n} x_i)a + (\sum_{i=1}^{n} y_i)b + nc = \sum_{i=1}^{n} \hat{u_i} \end{cases} \tag{34}$$

Let PU, PA, PB and PC be the coefficient matrices as follows:

$$PU = \begin{vmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} y_i^2 & \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} y_i & n \end{vmatrix} \tag{35}$$

$$PA = \begin{vmatrix} \sum_{i=1}^{n} x_i \hat{u_i} & \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} y_i \hat{u_i} & \sum_{i=1}^{n} y_i^2 & \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} \hat{u_i} & \sum_{i=1}^{n} y_i & n \end{vmatrix} \tag{36}$$

$$PB = \begin{vmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i\hat{u}_i & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} y_i\hat{u}_i & \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} \hat{u}_i & n \end{vmatrix} \tag{37}$$

$$PC = \begin{vmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} x_i\hat{u}_i \\ \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} y_i^2 & \sum_{i=1}^{n} y_i\hat{u}_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} y_i & \sum_{i=1}^{n} \hat{u}_i \end{vmatrix} \tag{38}$$

Because $PU \neq 0$, the parameters a, b and c have a unique solution, which is

$$\begin{cases} a = \frac{PA}{PU} \\ b = \frac{PB}{PU} \\ c = \frac{PC}{PU} \end{cases} \tag{39}$$

Previously, we proposed a quadratic regression function to predict partners' behavior, and also specified how to determine parameters a, b and c. However, it should be mentioned that the proposed quadratic regression function can only provide an *estimation* on possible partner behaviors, which might not exactly accord with the partners' *real* behaviors. In practice, agents' estimated behaviors should be close to their real actions. The closer the estimated behaviors to the real actions, the higher the probability that the estimated behaviors will occur. Thus we can deem that the differences (ε) between the estimation behaviors and the real behaviors obey the Gaussian distribution $N(\varepsilon, \sigma^2)$. Thus, if the deviation σ^2 can be calculated, we can make a precise decision on partner behaviors. It is known that there is more than 68% probability that partners' expected behaviors are located in the interval $[u-\sigma, u+\sigma]$, more than 95% that partners' expected behaviors lie in $[u-2\sigma, u+2\sigma]$, and more than 99% in the interval $[u-3\sigma, u+3\sigma]$.

In order to calculate the deviation σ, we firstly calculate the distance between the estimation results (u_i) and the real results on partners' behaviors ($\hat{u}_i$) as follows:

$$d_i = \hat{u}_i - u_i \tag{40}$$

It is known that all d_i ($i \in [1, n]$) obey the Gaussian distribution $N(0, \sigma^2)$. Then σ can be calculated as follows:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (d_i - \overline{d})^2}{n}} \tag{41}$$

where,

$$\overline{d} = \frac{1}{n} \sum_{i=1}^{n} d_i \tag{42}$$

Now by employing the Chebyshev inequality, we can calculate (1) the interval of partners' behaviors according to any accuracy requirements; and (2) the probability that any particular behavior may occur in potential partners in the future.

The Chebyshev's inequality is:

$$P(|\hat{u}_i - u_i| \geq \varepsilon) \leq \frac{\sigma^2}{\varepsilon^2} \tag{43}$$

where $\hat{u}_i$ is an instance, u_i is the mathematical expectation, σ is the deviation and ε is the accuracy requirement. This function indicates that based on a particular accuracy requirement ε, the possibility that the real behavior $\hat{u}_i$ is included in the interval $[u_i - \sigma, u_i + \sigma]$ is greater than $1 - \frac{\sigma^2}{\varepsilon^2}$.

In this chapter, we demonstrate three scenarios to indicate the agent behaviors prediction approach. Also, we compare the proposed quadratic regression approach with the Tit-For-Tat [9] and random approaches. The experimental results illustrate the outstanding performance of our proposed approach. In order to simplify the implemented process, all agents in our experiment employ the NDF [11] negotiation strategy. The partners' behaviors cover all possible situations in reality, which are conceder, linear and boulware. In experiments, we use the average error (EA) to evaluate the experimental results. Let u_i be the predicted result in cycle i and $\hat{u}_i$ be the real instance in cycle i, then AE_i is defined as follows:

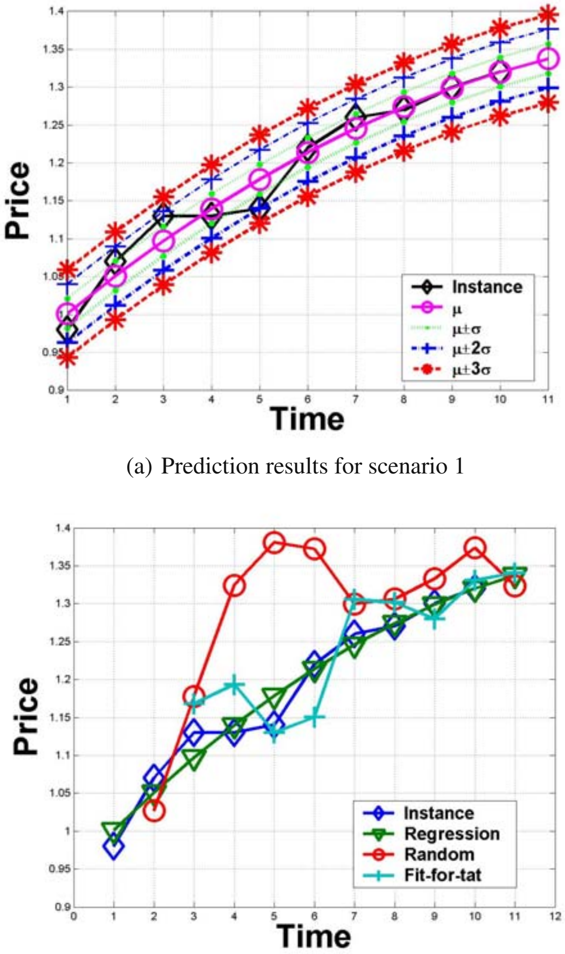
$$AE_i = \frac{\sum_{k=1}^{i} |\hat{u}_i - u_i|}{i} \tag{44}$$

The AE_i indicates the difference between the estimated results and the real value. The smaller the value of AE_i, the better the prediction result.

In the first scenario, a buyer wants to purchase a mouse pad from a seller. The acceptable price for the buyer is in $[\$0, \$1.4]$. The deadline for the buyer to finish this purchasing process is 11 cycles. In this experiment, the buyer adopts conceder behavior in the negotiation, and the seller employs the proposed approach to estimate the buyer's possible price in the next negotiation cycle. The estimated results are displayed in Figure 9(a) and the regression function is:

$$u = -0.002 * t^2 + 0.055 * t + 0.948$$

It can be seen that in the $8th$ negotiation cycle, the proposed approach estimates a price of \$1.26 from the buyer in the next cycle. Then according to the historical record in the $8th$ cycle, the real price given by the buyer in this cycle is \$1.26, which is exactly same as the estimation price. Furthermore, it can be seen that in cycles 4, 6, 9 and 10, the estimated prices are also the same as the real value. The estimation prices for the $2nd$, $3rd$ and $7th$ cycles are \$1.05, \$1.10 and \$1.25, respectively, and the real prices given by the buyer in these cycles are \$1.07, \$1.13, and 1.26, which differ only slightly between the estimated prices and real prices. According to Figure 9(a), all real prices are located in the interval of $[\mu - 2\sigma, \mu + 2\sigma]$, where μ is the estimated price and σ is the changing span. The $AE_{10} = 0.015$, which is only 1% of buyer's reserve price. Therefore, the prediction results by employing the proposed approach are very reliable.



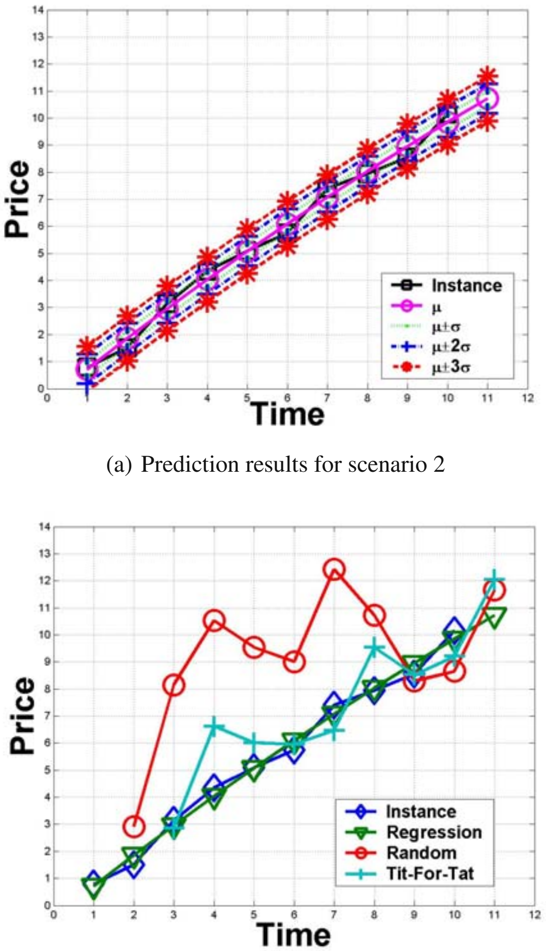
(a) Prediction results for scenario 1

(b) Prediction results comparison for scenario 1

Fig. 9 Scenario 1

In Figure 9(b), we compare results between the proposed approach and two other estimation approaches (Tit-For-Tat and random). It can be seen that even though the Tit-For-Tat approach can follow the trend of changes in the buyer's price, $AE_{10} = 0.078$ which is five times our proposed approach. For the random approach, it cannot even catch the main trend. The AE_{10} for the random approach is 0.11, which is ten times our proposed approach. The experimental results convince us that the proposed approach outperforms both the Tit-For-Tat and random approaches when a buyer adopts conceder negotiation behavior.

In the second scenario, a buyer wants to buy a keyboard from a seller. The desired price for the buyer is in the interval of $[\$0, \$14]$. We let the buyer employ the linear negotiation strategy, and still set the deadline to 11 cycles. The seller will employ



(a) Prediction results for scenario 2

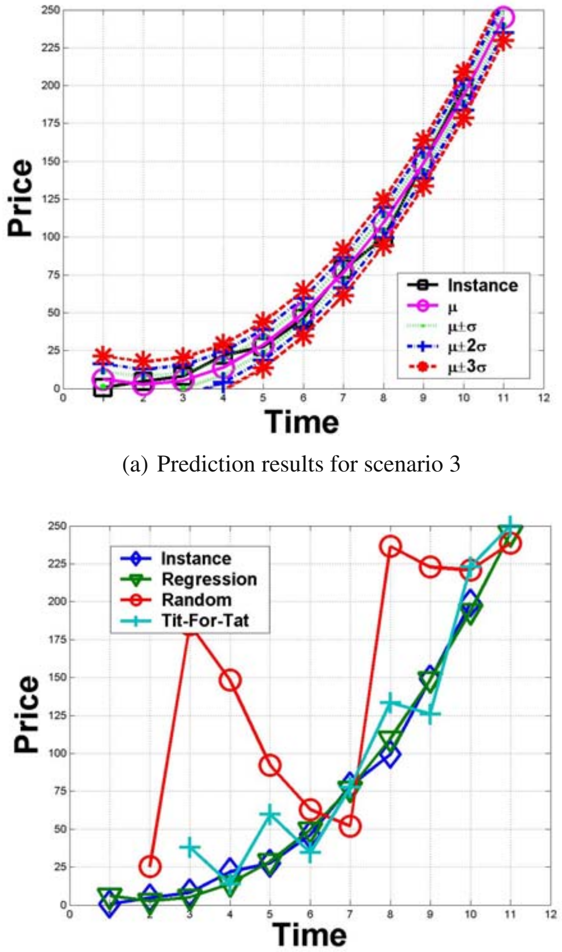
(b) Prediction results comparison for scenario 2

Fig. 10 Scenario 2

our proposed prediction function to estimate the buyer's offer. The estimated results are illustrated in Figure 10(a) and the estimated quadratic regression function is:

$$u = -0.015 * t^2 + 1.178 * t - 0.439$$

It can be seen that in the $3rd$, $5th$ and $8th$ cycles, the estimated prices are exactly the same as the real offers made by the buyer. The biggest difference between the estimated price and the real value is just 0.4, which happens in the $9th$ cycle. The average error in this experiment is only $AE_{10} = 0.24$, which is no more the 2% of the buyer's reserve price. The estimated quadratic regression function fits the real prices very well.



(a) Prediction results for scenario 3

(b) Prediction results comparison for scenario 3

Fig. 11 Scenario 3

Figure 10(b) compares results for the Tit-For-Tat approach, random approach and our proposed approach. It can be seen that the proposed approach is much closer to the real price than the other two approaches. The average error for the Tit-For-Tat approach is $AE_{10} = 2.52$, which is 18% of the buyer's reserve price. The average error for the random approach is very high – $AE_{10} = 4.82$ (34% of the buyer's reserve price). A second experimental result is that when partners perform with linear behaviors, the proposed approach also outperforms the other two approaches.

In the third scenario, a buyer wants to purchase a monitor from a seller. The suitable price for the buyer is in $[\$0, \$250]$. In this experiment, the buyer employs a boulware strategy in the negotiation. The deadline is still 11 cycles. The estimated quadratic function is:

$$u = 3.038 * t^2 - 12.568 * t + 15.632$$

The estimated results are shown in Figure 11(a), it can be seen that the proposed quadratic regression approach predicted buyer's prices successfully and accurately. Except for the $4th$ and $8th$ cycles, other estimated prices differ very little from the buyer's real offers. The average error in this experiment is $AE_{10} = 4.07$, which is only 1.6% of the buyer's reserve price. Therefore, we can say with confidence that from these estimation results, the seller can make accurate judgement about the buyer's negotiation strategy, and make reasonable responses in order to maximize its own benefit.

Finally, Figure 11(b) shows comparison results with two other estimation functions for the same scenario. For the Tit-For-Tat approach, the average error is $AE_{10} = 57.74$, which is 23% of the buyer's reserve price. For the random approach, the average error is $AE_{10} = 83.12$, which is 33% of the buyer's reserve price. Therefore, it can be seen that when the agent performs a boulware behavior, the proposed approach significantly outperforms the other two approaches.

From these experimental results, we can conclude that the estimated quadratic function regression approach can successfully estimate partners' potential behaviors. Moreover, the estimation results are accurate and sufficiently reasonable to be adopted by agents to modify their strategies in negotiation. The comparison results among the three types of agent behavior estimation also demonstrate the outstanding performance of our proposed approach.

In this section, we introduced agent negotiation for solving complex problems between collaborative agents. Firstly, we pointed out that agent competition can also be involved in collaborative problems. Then we introduced some basic knowledge about agent negotiation for conflict resolution. Furthermore, we introduced a partner selection approach and agents' behavior prediction approach for complex negotiation environments and illustrated some experimental results to show the improvements. In conclusion, we can say that agent negotiation is a very significant mechanism for agents to solve conflicts which may occur during complex problem solving procedures.

5 Conclusion

Complex problem solving requires diverse expertise and multiple techniques. MAS is a particularly applicable technology for complex problem solving applications. In a MAS, agents that possess different expertise and resources collaborate together to handle problems which are too complex for individual agents. Generally, agent collaborations in a MAS can be classified into two groups, namely agent cooperation and agent competition. These two kinds of collaborations are unavoidable for most MAS applications, but both present challenges. In addition, two main approaches for complex problem solving via agent cooperation and agent competition have been introduced – these being a dynamic team formation mechanism for cooperative agents, and a partner selection strategy for competitive agents. These two approaches can be applied to coordinate utility conflicts among agents, and make a MAS more suitable for open dynamic working environments.

Research into dynamic team formation can be extended in the following two directions. Currently, team formation research is based on a simple agent organisation. However, in many current MAS applications, more complex organisational structures, such as congregation [2], are adopted. Building a mechanism to support complex organisational formation is one research direction for the future. Furthermore, different organisational structures are suitable for different circumstances. In a complex dynamic working environment, agents may need to choose different organisational structures due to a changing environment. To develop mechanisms that enable agents to not only select cooperation partners but also dynamically choose organisational structures is another avenue for future research.

Further work on agent negotiation can proceed in two directions, as (i) Currently, most agent negotiation strategies and protocols can only handle the negotiation with single issue. However, with expansion of application domains, negotiating multiple issues will become a significant trend. Therefore, research on multi-issue negotiation will become a future direction. (ii) Most negotiation environments currently mainly focus on the static situations, but fail to take into account where a negotiation environment becomes open and dynamic. In an open and dynamic environment, agents can perform more flexibly to enhance their benefits. Also an open and dynamic negotiation environment is much more efficient in handling real world applications. Therefore, changing the negotiation environment from static to open and dynamic is another significant research direction on the topic of agent negotiation for the future.

Another potential direction is to extend our current research to complex domains in which agents can show semi-competitive behaviours or temporary collaborative behaviours in different situations.

Acknowledgements. The authors would like to acknowledge the support of the Intelligent Systems Research Centre at the University of Wollongong.

References

1. Artikis, A., Pitt, J.: A Formal Model of Open Agent Societies. In: Proceedings of the 5th International Conference on Autonomous Agents, Montreal, Canada, pp. 192–193 (2001)
2. Brooks, C., Durfee, E., Armstrong, A.: An Introduction to Congregating in Multiagent Systems. In: Proceedings of 4th International Conference on Multiagent Systems, Boston, USA, pp. 79–86 (2000)
3. Brzostowski, J., Kowalczyk, R.: On Possibilistic Case-Based Reasoning for Selecting Partners in Multi-agent Negotiation. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS, vol. 3339, pp. 694–705. Springer, Heidelberg (2004)
4. Candea, C., Hu, H., Iocchi, L., Nardi, D., Piaggio, M.: Coordination in Multi-agent RoboCup Teams. Robotic and Autonomous Systems 36(2), 67–86 (2001)
5. Castelpietra, C., Iocchi, L., Nardi, D., Piaggio, M., Scalzo, A., Sgorbissa, A.: Coordination among Heterogeneous Robotic Soccer Players. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Takamatsu, Japan, pp. 1385–1390 (2000)

6. Chajewska, U., Koller, D., Ormoneit, D.: Learning An Agent's Utility Function by Observing Behavior. In: Proceedings of 18th International Conference on Machine Learning, pp. 35–42. Morgan Kaufmann, San Francisco (2001)
7. Coehoorn, R., Jennings, N.: Learning on Opponent's Preferences to Make Effective Multi-issue Negotiation Trade-offs. In: Proceedings of the 6th International Conference on Electronic Commerce, ICEC 2004, Delft, Netherlands, October 2004, pp. 59–68. ACM Press, New York (2004)
8. Eberhart, R., Simpson, P., Dobbin, R.: Computational Intelligence PC Tools. AP Professional Press, Orlando (1996)
9. Faratin, P., Sierra, C., Jennings, N.: Negotiation Decision Functions for Autonomous Agents. Journal of Robotics and Autonomous Systems 24(3-4), 159–182 (1998)
10. Fatima, S., Wooldridge, M., Jennings, N.: Optimal Agendas for Multi-issue Negotiation. In: Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2003), pp. 129–136. ACM Press, New York (2003)
11. Fatima, S., Wooldridge, M., Jennings, N.: An Agenda-Based Framework for Multi-Issue Negotiation. Artificial Intelligence 152(1), 1–45 (2004)
12. Fatima, S., Wooldridge, M., Jennings, N.: Optimal Negotiation of Multiple Issues in Incomplete Information Settings. In: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), New York, USA, pp. 1080–1087. IEEE Computer Society, Los Alamitos (2004)
13. Gerkey, B., Mataric, M.: Multi-robot Task Allocation: Analyzing the Complexity and Optimality of Key Architectures. In: Proceedings of the IEEE International Conference on Robotics and Automation, Taibei, China, pp. 3862–3868 (2003)
14. Horling, B., Lesser, V.: A Survey of Multi-Agent Organizational Paradigms. The Knowledge Engineering Review 19, 281–316 (2005)
15. Ito, T., Parkes, D.: Instantiating the Contingent Bids Model of Truthful Interdependent Value Auctions. In: Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, pp. 1151–1159 (2006)
16. Jung, H., Tambe, M., Kulkarni, S.: Argumentation as Distributed Constraint Satisfaction: Applications and Results. In: Proceedings of the 5th International Conference on Autonomous Agents, Montreal, Canada, pp. 324–331. ACM Press, New York (2001)
17. Klir, G., Yuan, B. (eds.): Fuzzy Sets and Fuzzy Logic Theory and Applications. Prentic Hall, Upper Saddle River (1995)
18. Koes, M., Nourbakhsh, I., Sycara, K.: Heterogeneous Multirobot Coordination with Spatial and Temporal Constraints. In: Proceedings of the 20th National Conference on Artificial Intelligence (AAAI), June 2005, pp. 1292–1297. AAAI Press, Menlo Park (2005)
19. Kraus, S.: Strategic Negotiation in Multiagent Environments. The MIT Press, Cambridge (2001)
20. Lesser, V.: Reflections on the Nature of Multi-agent Coordination and Its Implications for an Agent Architecture. Journal of Autonomous Agents and Multi-Agent Systems 1(1), 89–111 (1998)
21. Lesser, V.: Cooperative Multiagent Systems: A Personal View of the State of the Art. IEEE Transactions on Knowledge and Data Engineering 11(1), 133–142 (1999)
22. Lesser, V., Horling, B., Klassner, F., Raja, A., Zhang, S.: BIG: An Agent for Resource-Bounded Information Gathering and Decision Making. Artificial Intelligence Journal, Special Issue on Internet Information Agents 118(1-2), 197–244 (2000)
23. Li, C., Chawla, S., Rajan, U., Sycara, K.: Mechanisms for Coalition Formation and Cost Sharing in an Electronic Marketplace. Technical Report CMU-RI-TR-03-10, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (April 2003)
24. Milanovic, N., Malek, M.: Current Solutions for Web Service Composition. IEEE Internet Computing Magazine 8(6), 51–59 (2004)

25. Munroe, S., Luck, M., d'Inverno, M.: Motivation-Based Selection of Negotiation Partners. In: 3rd International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS 2004, pp. 1520–1521. IEEE Computer Society, Los Alamitos (2004)
26. Nash, J.: The Bargaining Problem. Econometrica 18, 155–162 (1950)
27. Oliveira, E., Rocha, A.: Agents Advanced Features for Negotiation in Electronic Commerce and Virtual Organisations Formation Process. In: Sierra, C., Dignum, F.P.M. (eds.) AgentLink 2000. LNCS, vol. 1991, pp. 78–96. Springer, Heidelberg (2001)
28. Osborne, M., Rubenstein, A.: A Course in Game Theory. The MIT Press, Cambridge (1994)
29. Parsons, S., Sierra, C., Jennings, N.: Agents that Reason and Negotiate by Arguing. Journal of Logic and Computation 8(3), 261–292 (1998)
30. Rathod, P., desJardins, M.: Stable Team Formation among Self-interested Agents. In: Proceedings of AAAI Workshop on Forming and Maintaing Coalitions in Adaptive Multiagent Systems, San Jose, USA, pp. 29–36 (2004)
31. Ren, F., Zhang, M., Bai, Q.: A Fuzzy-Based Approach for Partner Selection in Multi-Agent Systems. In: Proceedings of the 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), Melbourne, Australia, pp. 457–462. IEEE Computer Society, Los Alamitos (2007)
32. Rigolli, M., Brady, M.: Towards a Behavioral Traffic Monitoring System. In: Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), Utrecht, Netherlands, pp. 449–454 (2005)
33. Rosenschein, J., Zlotkin, G.: Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers. MIT Press, Cambridge (1994)
34. Shen, J., Zhang, X., Lesser, V.: Degree of Local Cooperation and its Implication on Global Utility. In: Proceedings of 3rd International Joint Conference on Autonomous Agents and MultiAgent Systems, New York, USA, July 2004, vol. 2, pp. 546–553 (2004)
35. Sim, K.: A Survey of Bargaining Models for Grid Resource Allocation. ACM SIGECOM: E-commerce Exchange 5(5), 22–32 (2005)
36. Tambe, M.: Agent Architectures for Flexible, Practical Teamwork. In: Proceedings of the 14th National Conference on Artificial Intelligence, Rhode Island, USA, pp. 22–28 (1997)
37. Tambe, M.: Towards Flexible Teamwork. Journal of Artificial Intelligence Research 7, 83–124 (1997)
38. Tambe, M.: Implementing Agent Teams in Dynamic Multi-agent Environments. Applied Artificial Intelligence 12(2-3), 189–210 (1998)
39. Tsvetovat, M., Sycara, K.: Customer Coalitions in the Electronic Marketplace. In: Proceedings of the 4th International Conference on Autonomous Agents, Barcelona, Spain, pp. 263–264. ACM Press, New York (2000)
40. Yamamoto, J., Sycara, K.: A Stable and Efficient Buyer Coalition Formation Scheme for E-marketplaces. In: Proceedings of the 5th International Conference on Autonomous Agents, Montreal, Canada, pp. 576–583. ACM Press, New York (2001)
41. Zeng, D., Sycara, K.: Bayesian Learning in Negotiation. International Journal of Human-Computer Studies 48(1), 125–141 (1998)
42. Zhang, X., Lesser, V., Wagner, T.: Integrative Negotiation In Complex Organizational Agent Systems. In: Proceedings of the 1st International Joint Conference on Autonomous Agents & MultiAgent Systems (AAMAS 2002), July 2002, pp. 503–504 (2002)
43. Zhang, X., Lesser, V., Wagner, T.: Integrative Negotiation among Agents Situated in Organizations. IEEE Transactions on Systems, Man, and Cybernetics, Part C 36(1), 19–30 (2006)

Part V
Computer Vision

Predicting Trait Impressions of Faces Using Classifier Ensembles

Sheryl Brahnam and Loris Nanni

Abstract. In the experiments presented in this chapter, single classifier systems and ensembles are trained to detect the social meanings people perceive in facial morphology. Exploring machine models of people's impressions of faces has value in the fields of social psychology and human-computer interaction. Our first concern in designing this study was developing a sound ground truth for this problem domain. We accomplished this by collecting a large number of faces that exhibited strong human consensus in a comprehensive set of trait categories. Several single classifier systems and ensemble systems composed of Levenberg-Marquardt neural networks using different methods of collaboration were then trained to match the human perception of the faces in the six trait dimensions of intelligence, maturity, warmth, sociality, dominance, and trustworthiness. Our results show that machine learning methods employing ensembles are as capable as most individual human beings are in their ability to predict the social impressions certain faces make on the average human observer. Single classifier systems did not match human performance as well as the ensembles did. Included in this chapter is a tutorial, suitable for the novice, on the single classifier systems and collaborative methods used in the experiments reported in the study.

1 Introduction

Whenever we meet new people, we immediately form impressions of them. These impressions come from many sources: the social roles these people play, the

Sheryl Brahnam
Computer Information Systems, Missouri State University, 901 S. National,
Springfield, MO 65804, USA
e-mail: sbrahnam@missouristate.edu

Loris Nanni
DEIS, IEIIT—CNR, Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
e-mail: lnanni@deis.unibo.it

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 403–439.
springerlink.com

quality of their clothes and grooming, their height and weight, and the way they move, stand, and speak. One of the most important sources informing our initial assessments of people are the impressions we gather from their faces. As Cicero once remarked "Everything is in the face." The shape and texture of a face conveys information about a person's identity, gender, and age. Short term surface behaviors, such as eye blinking, direction of gaze, and facial gestures provide clues regarding a person's emotional and mental state, and the texture and color of the face are indicators of various health conditions.

Most modern people, however, would deny agreeing with Cicero's further claim that the face is inscribed with signs that reveal a person's inner essence and destiny. We are taught not to judge a book by its cover. Nonetheless, there is considerable evidence in recent studies in social psychology that people today are predisposed to form impressions of a person's social status, abilities, dispositions, and character traits based on nothing more than that person's facial appearance. When shown a face, not only are most people prepared to judge a person's gender, age, and emotional state, but also, as the social psychologist Bruce [12] has noted, that person's "personality traits, probable employment and possible fate" (p. 31). Furthermore, the evidence indicates that people are remarkably consistent and similar, across cultures and age groups, in their evaluations and reactions to faces [87].

Several theories have been advanced to explain why certain facial characteristics consistently elicit specific personality impressions. One major theory is that facial appearance is important because it guides people's behavior towards others that ensures the greatest chance of survival [56, 89]. Recognizing an angry face, for example, triggers lifesaving fight/flight responses. It is theorized that faces that are similar in structure to angry faces elicit similar responses. As Zebrowitz [87] explains, "We could not function well in this world if we were unable to differentiate men from women, friends from strangers, the angered from the happy, the healthy from the unfit, or children from adults. For this reason, the tendency to respond to the facial qualities that reveal these attributes may be so strong that it overgeneralizes to people whose faces merely resemble those who actually have the attribute" (pp. 14–15).

The most significant overgeneralization effects involve facial attractiveness, maturity, gender, and emotion. The morphological characteristics of these overgeneralization effects are associated with specific clusters of personality traits. Attractive people, for example, are associated with positive character traits. They are considered more socially competent, potent, healthy, intellectually capable, and moral than those less attractive. Attractive people are also perceived as being psychologically more adapted [47]. Facial abnormalities and unattractiveness, in contrast, elicit negative responses and are associated with negative traits [47]. Unattractive people are considered less socially competent and willing to cooperate [62]. They are also considered more dishonest, unintelligent, and psychologically unstable and antisocial. Unattractive people are often ignored and, if facially disfigured, avoided [16]. The unattractive are also more likely to be objects of aggression [3] and to suffer abuse [47].

Adults whose faces resemble those of babies (large eyes, high eyebrows, red lips that are proportionally larger, small nose and chin) are often treated positively as well [87], but they are also attributed childlike characteristics. Babyfaced people are perceived to be more submissive, naïve, honest, kindhearted, weaker, and warmer than others. They are also perceived as being more helpful, caring, and in need of protection [8]. Mature-faced individuals, in contrast, are more likely to command respect and to be perceived as experts [87].

The overgeneralization effect of gender is strongly correlated with facial maturity [87]. Female faces, more than male faces, tend to retain into adulthood the morphological characteristics of youth [28] and are more likely to be ascribed characteristics associated with babyfacedness: female faces are thought to be more submissive, caring, and in need of protection. Similarly, male faces, tending to be morphologically more mature, are perceived as having the psychological characteristics typically associated with mature-faced individuals: male faces are thought to be more dominant, intelligent, and capable.

While many social psychologists believe that facial impressions of character are related in part to the morphological configurations that characterize emotional displays, the overgeneralization effect of emotion has not received as much attention as some of the other overgeneralization effects. Nonetheless, there is evidence supporting the idea that morphological configurations suggestive of emotional expressions play a role in the formation of trait impressions. Take smiling for instance. People react positively to smiling faces and find them disarming and thus not very dominant [39]. A person whose lips naturally curl upward would thus be perceived more positively than a person whose lips tend to turn downward, as illustrated in Figure 1.

In the study reported in this chapter, we experiment with face recognition systems made of single classifiers and ensembles to see whether these techniques are capable

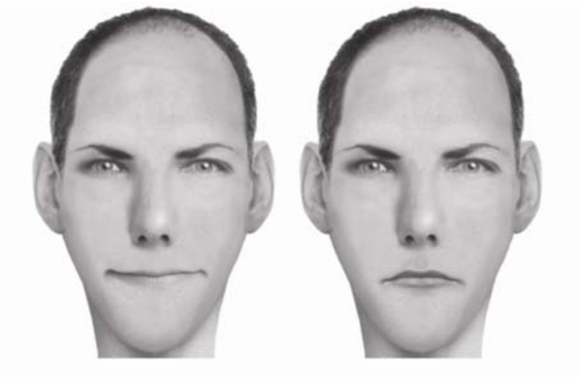


Fig. 1 Illustration of the overgeneralization effect of emotion. A face with lips that naturally turn upwards (left) is perceived similarly to smiling faces, that is, as low in dominance, whereas a face with lips that turn downwards (right) is perceived as more threatening. In the two images, only the lips differ. These faces were generated using *FACES* by InterQuest and Micro-Intel

of predicting the social meanings different facial morphologies produce. Face recognition involves the classification of facial morphology mostly in terms of identity [1]. Application areas where face recognition has been successfully employed include biometrics, information security, law enforcement, surveillance, and access control. Since gender, age, and race are closely tied to identification, these facial characteristics have also been the focus of many investigations [13, 49, 61, 64, 65, 85].

Our application differs significantly from most face classification tasks in that the ground truth is not based on factual information about the subjects but rather on the impressions faces make on the average observer. Few classification experiments have focused on the average observer's impressions of faces. One of the first was [9]. In that study, holistic face classifiers, based on principle component analysis (PCA), were trained to match the human classification of faces along the bipolar rating extremes of the following trait dimensions: adjustment, dominance, warmth, sociality, and trustworthiness. Although results were marginally better than chance in matching the perception of dominance (64%), classification rates were significantly better than chance for adjustment (71%), sociality (70%), trustworthiness (81%) and warmth (89%). The dataset used in that study, however, may have positively influenced classification rates. The faces were randomly generated using the full database of photographs of facial features (eyes, mouths, noses, and so forth) found in the popular composite software program FACES [30], produced by InterQuest and Micro-Intel. Although it was possible to generate a fairly large number of unique faces by randomly manipulating individual facial features in the FACES database, subject ratings of the faces produced small classes (less than 15 faces each) that were strongly associated with the bipolar extremes for each trait dimension. We believe that the low number of faces in each trait class inflated recognition rates.

Related to our work are face recognition experiments that have recently been performed to detect facial attractiveness. In [38], for instance, faces are compared to an archetypical mask with good recognition rates. Two sets of photographs of 92 Caucasian females, approximately 18 years old, were rated by subjects using a 7-point scale. In their experiments, they classified faces into two classes, *attractive* (highest 25% rated images) and *unattractive* (lowest 25% rated images). Performance, based on percentage of correctly classified images, averaged 76.0% using K-nearest neighbor and 70.5% using linear Support Vector Machines. The authors believed that the low number of faces in their two classes of *attractive* and *unattractive* faces (numbering approximately 24 images each) accounted for the poor performance.

As illustrated in both these studies, a major problem with modeling human attributions of faces, whether along a number of trait dimensions or in terms of attractiveness, is developing a sufficiently large database of representative faces. One of the contributions of the study reported in this chapter and discussed in Section 3 is the production of six databases of faces that were found to be strongly associated with the bipolar extremes of the following six trait dimensions: intelligence, maturity, dominance, sociality, trustworthiness, and warmth. To create these databases, 480 stimulus faces were artificially constructed by four artists using FACES. The artists were instructed to produce faces they felt would be perceived as either high or low in the six trait dimensions. Using this method, we were able to produce trait classes that averaged 111 faces.

A variety of approaches are available to deal with such complex pattern recognition problems as face recognition. Typically the goal is to find the best single classifier for the task. In most application areas, holistic algorithms are preferred because they are easy to implement and have been shown to outperform other methods [48]. In addition, they allow the classifier to determine the best set of features available in the raw pixel data of the images. Holistic algorithms, however, suffer from what is commonly referred to as the *curse of dimensionality* [6]. The pixel values of image files contain a very small number relevant to the classification problem. Processing large numbers of insignificant values greatly increases computational complexity and degrades performance. For this reason, most holistic classification systems apply various methods to reduce dimensionality. In Section 2, we describe several holistic face recognition algorithms as well as the general principles behind feature reduction and extraction.

A well established technique for improving single classifier performance is based on building ensembles from classifiers that perform less optimally as a whole but which nonetheless contribute some essential information. Ensembles have been shown to achieve higher classification rates than single classifier systems [37, 40]. As described in some detail in Section 2, the diversity of classifier evaluations, augmented by such collaborative methods as boosting and bagging [10] and the availability of combination rules [23] are major reasons for improved performance. The superiority of collaborative methods is borne out by the results of our study reported in Section 4. Ensembles composed of 100 Levenberg-Marquardt neural networks (LMNNs) using different methods of collaboration proved to be as capable as most individual human beings are in their ability to predict the social impressions certain faces make on the average human observer. Single classifier systems, including systems using a single LMNN, did not match human performance as well as the ensembles did.

We believe that exploring machine models of people's impressions of faces has value in several fields, most notably in social psychology and human-computer interaction. In psychology, building models of human perception could expand our understanding of the specific characteristics of faces that impact human social impressions. Depending on the types of classifier systems used (ensembles of neural networks for instance), it may be possible to build representative models of the brain's processing of faces for impression formation.

A major area of research in human-computer interaction (HCI) involves building socially intelligent interfaces. A human-like interface capable of matching the average observer's impressions of a user's face could use this information to determine an initial interaction strategy that is socially adept. The interface could assume that users have been treated in ways that reflect the trait impressions of their faces. People that look dominant, for instance, probably feel comfortable with an initial reaction that shows some deference, whereas commenting on a person's intelligence might best be avoided altogether if the user appears unintelligent. A system that predicts how others view people could also help the human-like interface talk more intelligently with users about other people, such as celebrities. Other application areas of a trait prediction system could include the development of smart mirrors that inform people how other's might view them. This could assist people in composing their faces for various social settings, such as job interviews. It may also be possible to extend social perception systems of faces to include other aspects of human appearance and behavior.

The remainder of this chapter is organized as follows. In Section 2, we provide a tutorial suitable for the novice on such basic classifiers in computer vision as PCA and nearest neighbor (NN), support vector machines (SVMs) [84], enhanced subspace methods (SUB) [66], and Levenberg-Marquardt neural networks (LMNN) [24]. We then define methods for creating classifier ensembles using multiple LMNNs for enhanced performance. In particular, we describe such collaborative methods as bagging (BA), random subspace (RS), and class switching (CW). In Section 3, we present the study design and our method of generating the stimulus faces and evaluating subject ratings of the faces. In Section 4, we present our classifier systems and compare the performance of some simple classifiers (PCA+NN, SUB, SVM, and LMNN) to classifier ensembles (100 LMNN selected using RS, BA and CW). Ensembles of LMNN perform only slightly better than the best single classifier (SVM), but the ensembles are more stable in performance across all six trait dimensions and, as already noted, more closely approximate the performance of individual human raters. We conclude this chapter in Section 5 by noting some of the contributions and weaknesses of the study reported in this chapter. We also mention our current work developing a larger database of photographs of people that strongly produce specific trait impressions.

Although Section 2 provides a fairly comprehensive tutorial of the single classifier systems and collaborative methods used in building our ensembles, we recommend reading Kuncheva's book [44] for additional details on developing ensembles using a larger variety of collaborative methods. For general books on machine learning, pattern recognition and classification, Alpaydin [4], Duda, Hart and Stork [24], and Russell and Norvig [73] provide particularly good complementary overviews.

2 Face Classification: Single Classifier Algorithms and Collaborative Methods

Face classification can be defined as the computer vision problem of assigning predefined labels to samples of one or more faces. One of the first considerations when developing a face classification system is deciding on a method for representing faces. A method used in many early studies involved measuring the relative distances between important facial key points: eye corners, mouth corners, nose tip, and chin edge. Although this approach has the advantage of drastically reducing the number of features to be considered, a major drawback has been the difficulty in determining the best set of key points to measure [17, 82]. An alternative approach is to process faces holistically [14]. Holistic techniques discover the features

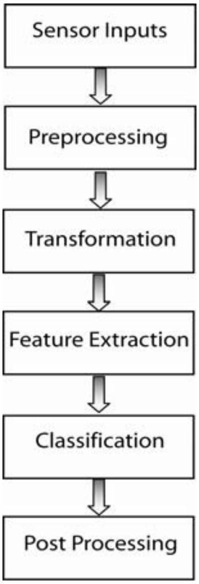


Fig. 2 General schematic of holistic face classification systems

relevant to a classification problem and have been shown to outperform classification systems that rely on facial key points [48].

In holistic systems, face images are represented as single points in a high dimensional vector space. As illustrated in Figure 2, input into the system is originally collected by a sensor. In most face recognition tasks, the sensor inputs are *m* x *n* pixel values. In the preprocessing stage these values are filtered of noise and undergo various normalization processes, such as illumination correction, subtraction of the mean, and division of each value by the variance. The resulting values are then concatenated into N feature vectors, $\mathbf{x}$, forming a data set, $\mathbf{X} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_N^T]$, where each feature vector $\mathbf{x}$ belongs to R^D.

Because of the large size of the feature vectors, processing each vector results in high computational costs. To contend with the *curse of dimensionality* [6], the original feature space is generally transformed so that the most relevant variables can be extracted. Feature transformation can occur within the spatial domain using statistical methods, such as principle component analysis (PCA) and linear discriminant analysis (LDA), or within the frequency domain using such methods as the discrete Fourier transform, Gabor wavelet transform, and discrete cosine transform.

The basic principle of a transform is to take the preprocessed feature vector and arrive at another vector of rotated values. Although the two sets of values carry the same amount of energy and information, the transforms tend to decorrelate the features and redistribute most of the energy contained in the raw input into a smaller number of components, thus allowing for *feature extraction*. In feature extraction, the feature set is greatly reduced by selecting only those features that possess the most significant information.

In the classification stage, the extracted feature vectors are divided into training and testing sets. Classifiers, such as the nearest neighbor (NN) [21], artificial neural networks (ANNs), support vector machines (SVMs) [84] and Oja's subspace (SUB) [66] are then given the task of learning to map the training vectors to a set of labels, or classes.

In practice, a simple 1-NN classifier is used to benchmark the performance of other classifiers, since it performs well in most applications and involves few user-controlled parameters. NN is usually employed in all techniques that adopt dimensionality reduction (e.g., PCA). Classification in this case involves projecting an unknown face vector onto the transformed face space and measuring its distance from representative classes within the same space. PCA has successfully been used to classify faces according to identity [75, 78, 79], gender [35, 65, 80], age [82], race [52, 64], and facial expression [20, 55, 69].

ANNs are modeled after biological neurons. They are massively parallel computing systems composed of simple processors that are highly interconnected. At its simplest, a neuron, or node, in the system is given an input, *i*, that is transformed by a weight, *w*. Node output is then dependent on a transfer function. The central idea of an ANN is to adjust architecture parameters in the training process in such a way that the network learns to correctly map the training vectors to their assigned classes. Learning algorithms (based mostly on a form of gradient descent) search through the problem space to find a function that solves this problem with the lowest cost.

ANNs have proven capable of handling a variety of face recognition tasks: gender classification [26, 31, 59], face identification, [19] face detection [72], and facial expression recognition [69]. Kohonen [41] was one of the first to use a linear autoassociative neural network to store and recall face images. Autoassociative neural networks associate input patterns with themselves [81]. It is interesting to note that a linear autoassociative neural network is equivalent to PCA [67]. Early surveys of neural network face classification techniques can be found in [18] and [82].

SVMs are powerful binary classifiers. They determine a decision boundary in the feature space by constructing the optimal separating hyperplane that distinguishes the classes. Using SVMs to classify faces is a recent development [33, 60, 70], yet SVMs have already established a proven track record [18, 70, 92]. They typically outperform PCA [32, 33, 60] and provide performance comparable to neural networks.

Once classifiers have been trained, the measure of classifier performance is the error rate produced when presented the testing set of samples. In the post-processing stage, various correction techniques are implemented and scores are normalized. The percentage of misclassified test samples provides an estimate of the error rate. For this estimate to be reliable, the training and testing sets should be sufficiently large. In practice this is often not the case, and a small training set results in classifiers that do not generalize well. If the test set is small, then the confidence levels are also low. Cross-validation approaches, e.g., the leave-one-out and rotation schemes, strengthen classifier generalizability and confidence levels. However, there are limitations to these approaches. A bootstrap method proposed by [27] has been shown to outperform cross-validation approaches by resampling training patterns with replacement to generate a large number of artificial training sets [36].

The Receiver Operating Characteristic (ROC) Curve, which plots the False Reject Rate (FRR) versus the False Acceptance Rate (FAR) assesses system performance at various operating points [58]. FAR is the ratio of the number of incorrectly matched patterns of different classes to the total number of attempted matches. FRR is the ratio of the number of incorrectly matched patterns of the same class to the total number of attempted matches. ROC provides a more comprehensive measure of the error rate than that provided by FAR and FRR alone. The most commonly reported version of the ROC curve is the area under the ROC curve, or AROC.

Typically error rates are used to find the best single classifier for the face classification task. However, much recent work has focused on employing ensembles of classifiers. Despite new advancements in the design of single classifiers, ensembles continue to achieve higher classification rates than single classifier systems [37, 40]. This good performance has been demonstrated both theoretically and empirically [5, 40, 68], especially when ensembles are built using classifiers that are both accurate and independent, i.e., when the classifiers make errors on different regions of the feature space [57, 86, 91].

Figure 3 illustrates the basic framework for training a classifier ensemble. In step 1, multiple datasets are created from the master training set. In step 2, a set of base classifiers are trained on the subsets. In step 3, classifier opinions are combined in some fashion using a combiner, C^*. The basic idea is to construct a set of classifiers from the training data and to predict class labels from previously unseen records in the testing set by aggregating predictions made by the ensemble.

Some of the most popular methods for assembling classifier ensembles are Bagging (BA), Random Subspace (RS), and Class Switching (CW). In bagging [11], a randomly selected subset of the training set is extracted (with replacement) for training each base classifier. In Random Subspace [34], a randomly selected feature subset of the original feature set is used for training each base classifier. In class switching [54], the classes of the training samples are randomly switched according to a given probability that is based on the original training set.

There are several methods for combining classifier results. The most typical combination methods are majority voting, sum rule, max rule, min rule, product rule, median rule, and Borda count. The sum rule (averaging) has been shown to outperform other classifier combination rules and is more resilient to estimation errors [40]. Voting, the sum rule, and Borda count are static, with no training required, while others are trainable. The trainable combiners work better if a large training set is available for training the combiners.

Ensemble performance depends on the diversity of the classifiers. Different metrics have been proposed in the literature to evaluate whether the individual classifiers in the ensemble are independent from each other. The most reliable and frequently used metric is the Yule's Q-statistic [43, 45]. For two classifier D_i and D_j the Q-statistic is an *a posteriori* measure that is defined as:

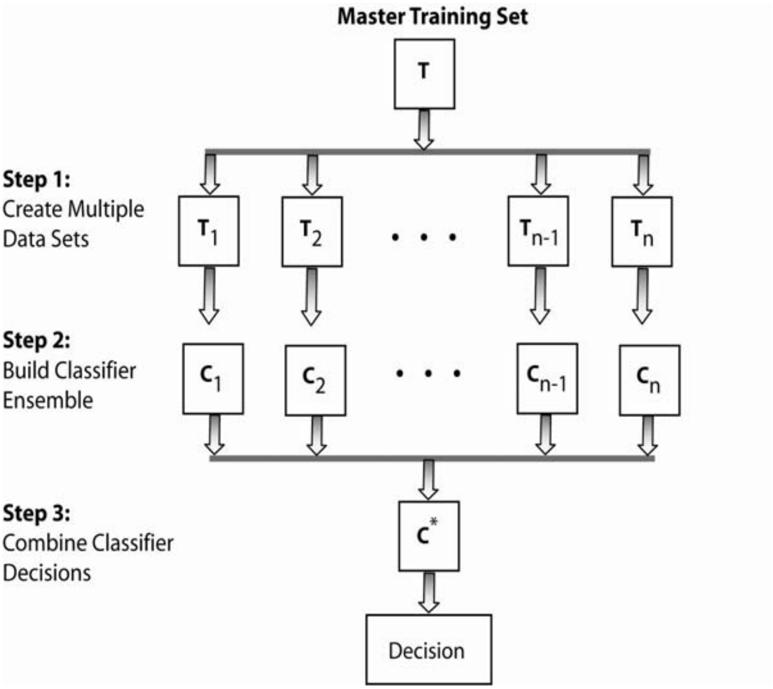


Fig. 3 General schematic of classifier ensembles

$$Q_{i,j} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} \tag{1}$$

where N^{ab} is the number of instances in the test set, classified correctly (a=1) or incorrectly (a=0) by the classifier D_i, and correctly (b=1) or incorrectly (b=0) by the classifier D_j. Q varies between -1 and 1, and it is 0 for statistically independent classifiers. Classifiers that tend to recognize the same patterns correctly will have $Q > 0$, and those which commit errors on different patterns will have $Q < 0$.

Readers wanting a more comprehensive survey of face classification techniques should refer to [92] for single classifier systems and [37] for more recent advances using classifier ensembles. The remainder of this section provides a basic tutorial on the classification algorithms used in the experiments presented in this chapter. In Section 2.1, we describe PCA (as a transform and classification method using NN), SVM, Oja's subspace (SUB), and the Levenberg-Marquardt neural network (LMNN) algorithm for computing gradient descent in feedforward neural networks. This is followed by a description in Section 2.2 of the methods we used to build classifier ensembles, *viz.*, bagging (BA), class switching (CW), and random subspace (SUB). We conclude the tutorial in Section 2.3 by listing software resources that implement the algorithms discussed in this section and by listing some excellent general books on machine learning and pattern recognition and classification.

2.1 Single Classifier Algorithms

In this Subsection we provide a general discussion, followed by an algorithmic outline, of each of the classifiers used in this study: PCA and NN, SUB, SVM, and LMNN.

2.1.1 Principle Component Analysis (PCA)

The central idea behind PCA is to find an orthonormal set of axes pointing in the direction of maximum covariance in the data. In terms of facial images, the idea is to find the orthonormal basis vectors, or the eigenvectors, of the covariance matrix of a set of images, with each image treated as a single point in a high dimensional space. It is assumed that facial images form a connected subregion in the image space. The eigenvectors map the most significant variations between faces and are preferred over other correlation techniques that assume every pixel in an image is of equal importance, (see, for instance, [42]). Since each image contributes to each of the eigenvectors, the eigenvectors resemble ghostlike faces when displayed. For this reason, they are often referred to in the literature as *holons* [19] or *eigenfaces* [78], and the new coordinate system is referred to as the *face space* [78]. Examples of eigenfaces are shown in Figure 4. Individual images can be projected onto the face space and represented exactly as weighted combinations of the eigenface components, as illustrated in Figure 5.

The resulting vector of weights that describes each face can be used both in face classification and in data compression. Classification is performed by projecting a

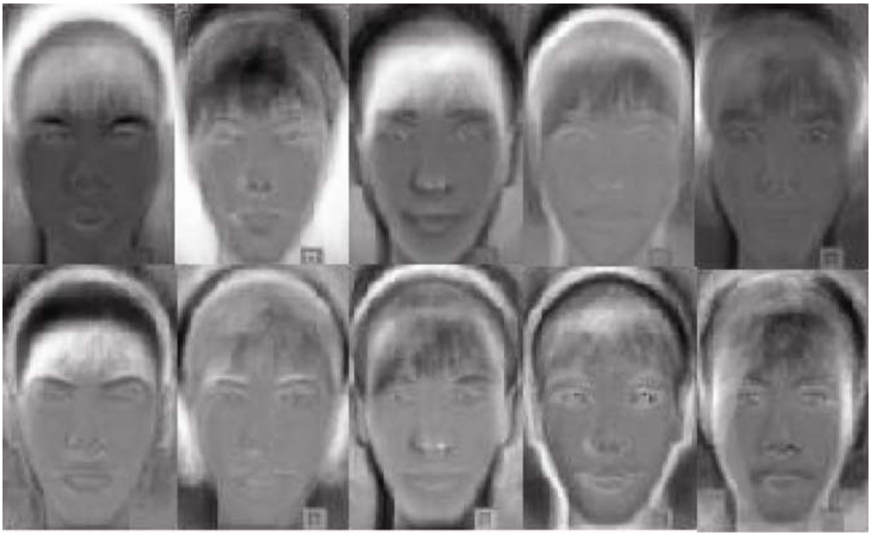


Fig. 4 The first ten eigenfaces of the 480 stimulus faces generated for our experiments, with the eigenfaces ordered by magnitude of the corresponding eigenvalue

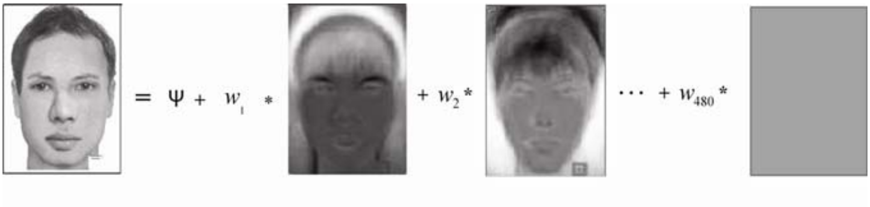


Fig. 5 Illustration of the linear combination of eigenfaces. The face to the left can be represented as a weighted combination of eigenfaces plus ψ the average face (see Equation 3)

new image onto the face space and comparing the resulting weight vector to those of a given class [78]. Compression is achieved by reconstructing images using only those few eigenfaces that account for the most variability [74]. PCA classification and compression are discussed in more detail below.

PCA Classification

The principal components of a set of images can be derived directly as follows. Let $\mathbf{I}(x,y)$ be a two-dimensional array of intensity values of size NxN. $\mathbf{I}(x,y)$ may also be represented as a single point, a one-dimensional vector $\boldsymbol{\Gamma}$ of size N^2. Let the set of face images be $\boldsymbol{\Gamma}_1, \boldsymbol{\Gamma}_2, \boldsymbol{\Gamma}_3, \ldots \boldsymbol{\Gamma}_{\mathrm{M}}$, and let

$$\boldsymbol{\Phi}_k = \boldsymbol{\Gamma}_k - \boldsymbol{\Psi} \tag{2}$$

represent the mean normalized column vector for a given face $\boldsymbol{\Gamma}$, where

$$\boldsymbol{\Psi} = \frac{1}{M}\sum_{k=1}^{M} \boldsymbol{\Gamma}_k \tag{3}$$

is the average face of the set. PCA seeks the set of M orthonormal vectors, $\mathbf{u}_{\mathrm{k}}$, and their associated eigenvalues, λ_{k}, which best describes the distribution of the image points. The vectors $\mathbf{u}_{\mathrm{k}}$ and scalars λ_{k} are the eigenvectors and eigenvalues, respectively, of the covariance matrix:

$$\mathbf{C} = \frac{1}{M}\sum_{k=1}^{M} \boldsymbol{\Phi}_k \boldsymbol{\Phi}_k^T = \mathbf{A}\mathbf{A}^T \tag{4}$$

where the matrix $\mathbf{A} = [\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2, \ldots, \boldsymbol{\Phi}_{\mathrm{M}}]$ [78].

The size of $\mathbf{C}$ is N^2 by N^2 which for typical image sizes is an intractable task [78]. However, since typically $M < N^2$, that is, the number of images is less than the dimension, there will only be $N-1$ nonzero eigenvectors. Thus, the N^2 eigenvectors can be solved in this case by first solving for the eigenvectors of an M x M matrix, followed by taking the appropriate linear combinations of the data points $\boldsymbol{\Phi}$ (see [78]).

PCA is closely associated with the singular value decomposition (SVD) of a data matrix. SVD can be defined as follows:

$$\boldsymbol{\Phi} = \mathbf{USV}^T \tag{5}$$

where $\mathbf{S}$ is a diagonal matrix whose diagonal elements are the singular values, or eigenvalues, of $\boldsymbol{\Phi}$, and $\mathbf{U}$ and $\mathbf{V}$ are unary matrices. The columns of $\mathbf{U}$ are the eigenvectors of $\boldsymbol{\Phi\Phi}^T$ and are referred to as *eigenfaces*. The columns of $\mathbf{V}$ are the eigenvectors $\boldsymbol{\Phi\Phi}^T$ and are not used in this analysis.

Faces can be classified by projecting a new face $\boldsymbol{\Gamma}$ onto the face space as follows:

$$\omega_k = \mathbf{u}_k^T \left(\boldsymbol{\Gamma}_k - \boldsymbol{\Psi}\right) \tag{6}$$

for k= 1, $\ldots M$ eigenvectors. The weights form a vector $\boldsymbol{\Omega}_k^T = [\omega_1, \omega_2, \ldots, \omega_M]$, which contains the projections onto each eigenvector. Classification is performed by calculating the distance of $\boldsymbol{\Omega}_k$ from $\boldsymbol{\Omega}$, where $\boldsymbol{\Omega}$ represents the average weight vector defining some class [78].

Two commonly used distance measures are the sum of absolute differences, also known as the L$_1$ metric, and the Euclidean distance, also know as the L$_2$ metric. If we have 2 points, A $(x_{1,}y_1)$ and B $(x_{2,}y_2)$, the L$_1$ distance between A and B is $abs(x_1 - x_2)$ + abs $(y_1 - y_2)$. The L$_2$ metric is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

PCA Representation

Since the eigenfaces are ordered, with each one accounting for a different amount of variation among the faces, images can be reconstructed using only those few eigenfaces, $M' << M$ in Equation 4, that account for the most variability [74]. As noted above, PCA results in a dramatic reduction of dimensionality and maps the most significant variations in a dataset. For this reason, it is often used to transform and reduce features when performing other classification procedures. In the experiments reported in this chapter, we retain those components that account for 0.98 of the variance.

Outline PCA

The basic steps necessary to perform PCA training and testing using face images are outlined in Table 1. These steps follow from the presentation given above.

2.1.2 Support Vector Machines (SVMs)

Support Vector Machines (SVMs), introduced in [84], belong to the class of maximum margin classifiers. They perform pattern recognition between two classes by finding a decision surface that has maximum distance to the closest points in the training set. The data points that define the maximum margin are called support vectors.

Table 1 Outline of PCA Training and Testing Steps

PCA Training	PCA Testing
Using a set of training feature vectors: 1. Compute the average feature vector, $\boldsymbol{\Psi}$; 2. Subtract $\boldsymbol{\Psi}$ from the feature vectors to obtain $\boldsymbol{\Gamma}$, the mean adjusted dataset; 3. Derive eigenfaces for $\boldsymbol{\Gamma}$ using SVD; 4. Obtain the weight vectors, or eigenvalues, for each $\boldsymbol{\Gamma}_k$ by projecting it onto the resulting face space; 5. Reduce dimensionality by retaining only the most significant eigenvalues; 6. Obtain the class vectors, $\boldsymbol{\Omega}$, by averaging the eigenvalues of each $\boldsymbol{\Gamma}_k$ belonging to each class.	Using a set of testing feature vectors: 1. Subtract $\boldsymbol{\Psi}$ from the feature vectors to obtain $\boldsymbol{\Gamma}$; 2. Obtain the weight vectors, or the eigenvalues, for each $\boldsymbol{\Gamma}_k$ by projecting $\boldsymbol{\Gamma}_k$ onto the face space derived using the training set; 3. Reduce dimensionality as was done with the training set; 4. Classify each $\boldsymbol{\Gamma}_k$based on its distance from $\boldsymbol{\Omega}$, using a distance metric.

SVMs are designed to solve two-class problems. SVMs produce the pattern classifier 1) by applying a variety of kernel functions (linear, polynomial, radial basis function, and so on) as the possible sets of approximating functions, 2) by optimizing the dual quadratic programming problem, and 3) by using structural risk minimization as the inductive principle, as opposed to classical statistical algorithms that maximize the absolute value of an error or of an error squared.

Different types of SVM classifiers are used depending upon the type of input patterns: a linear maximal margin classifier is used for linearly separable data, a linear soft margin classifier is used for linearly nonseparable, or overlapping, classes, and a nonlinear classifier is used for classes that are overlapped as well as separated by nonlinear hyperplanes. Each of these cases is outlined below. Readers interested in using SVM should consult [22].

Outline of SVM

Suppose, there is a set of training data, $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$ where $\mathbf{x_i} \in \mathbf{R^n}$ and $i = 1, 2, \ldots, k$. Each $\mathbf{x_i}$ belonging as it does to one of two classes, has a corresponding value y_i where $y_i \in \{-1, 1\}$.

Linear maximal margin classifier

The goal is to build the hyperplane that maximizes the minimum distance between the two classes. This hyperplane is called the Optimal Separating Hyperplane (OSH) and has the form:

$$f(\mathbf{x}) = \sum_{i=1}^{k} \alpha y_i \mathbf{x}_i \cdot \mathbf{x} + b \tag{7}$$

where α and b are the solutions of a quadratic programming problem. Unknown test data $\mathbf{x}_t$ can be classified by simply computing Equation 8.

$$f(x) = sign(\mathbf{w}_0 \bullet \mathbf{x}_t + b_0) \tag{8}$$

Examining Equation 8, it can be seen that the hyperplane is determined by all the training data, $\mathbf{x}_i$ that have the corresponding attributes of $\alpha_i > 0$. We call this kind of training data *support vectors*. Thus, the optimal separating hyperplane is not determined by the training data per se but rather by the support vectors.

Linear soft margin classifier

The objective in this case is to separate the two classes of training data with a minimal number of errors. To accomplish this, some non-negative slack variables, $\xi_{i,i}=$ $1,2,\ldots\ k$, are introduced into the system. The penalty, or regularization parameter, C, is also introduced to control the cost of errors. The computation of the linear soft margin classifier is the same as the linear maximal margin classifier. Thus, we can obtain OSH using Equations 7 and 8.

Nonlinear classifier

In this case, kernel functions such as the polynomial or RBF are used to transform the input space to a feature space of higher dimensionality. In the feature space, a linear separating hyperplane is sought that separates the input vectors into two classes. In this case, the hyperplane and decision rule for the nonlinear training pattern is Equation 9.

$$f(x) = sign(\sum_{i=1}^{K} \alpha_i y_i \mathbf{K}(\mathbf{x}_t, \mathbf{x}) + b) \tag{9}$$

Where, α_i and b are the solutions of a quadratic programming problem and K($\mathbf{x}_{t,}\mathbf{x}$) is a kernel function.

2.1.3 Oja's Subspace Learning and Classification (SUB)

In PCA we are approximating images as a linear combination of some subset of orthogonal basis vector, the eigenfaces. Each choice of orthonormal vectors gives a different approximation. There are many alternative criteria for choosing a set of basis vectors. Oja [66] lists six:

1. Minimum approximation error (least average distance between x and $\hat{x}$);
2. Least representation entropy;
3. Uncorrelated coefficients (with each other);
4. Maximum variances of the coefficients;
5. Maximum separation of two classes in terms of distance in a single subspace;
6. Maximum separation of two classes in terms of approximation errors in two subspaces.

Number 1 gives rise to the Karhunen-Loeve Transform and number 2 to PCA. The outline below presents the operations needed when dealing with subspaces (numbers 4 and 5 above).

Outline of SUB

The algorithm for the creation of the subspace related to each class is divided into two phases:

1. Normalization: all the objects in each class are normalized such that their Euclidian distance to the origin is one. This normalization is useful for employing the norm of the projection of a pattern on a subspace as similarity measure;
2. Subspace generation: for each class, a PCA subspace is calculated (see the PCA section for details).

The algorithm for the classification of each test image is divided in two phases:

1. Projection of the pattern: a map between the original space and the reduced eigenspace is performed by means of the operator of projection (see the PCA section for details);
2. Distance calculation: the norm of the projection of a pattern on each subspace is used as similarity measure between the input vector and the class related to the subspace. The input vector is then classified according to the maximal similarity value $argmax_{j=1,\dots,s}[\log(\|\mathbf{y}_i\|^2) - \log(1 - \left\|\mathbf{y}_j\right\|^2)]$, where s is the number of classes and $\mathbf{y}_j$ is the vector $\mathbf{x}$ projected onto the space of the class j.

2.1.4 Levenberg-Marquardt Neural Networks (LMNN)

The LMNN algorithm was first presented in [50]. Marquardt [53] rediscovered the algorithm approximately twenty years later. It is now a widely used optimization algorithm, solving the problem of nonlinear least squares minimization using Gauss-Newton's iteration in combination with gradient descent. It is considered one of the fastest methods for training moderate sized feedforward neural networks. For a more comprehensive tutorial see [63].

Outline of LMNN

Gradient descent is a simple method for finding the minima in a function but suffers from a number of convergence problems. When the gradient is small, intuitively it makes sense to take large steps down the gradient. Conversely, when the gradient is large, it would be logical to take smaller steps. The exact opposite occurs in gradient descent, which updates a parameter at each step by adding a negative of the scaled gradient:

$$x_{i+1} = x_i - \lambda \nabla f \tag{10}$$

By using the second derivative and by expanding the gradient of f using a Taylor series, Equation 10 can be improved as follows:

$$\nabla f(x) = \nabla f(x_0) + (x - x_0)^T \nabla f(x_0) + \text{higher order terms of} (x - x_0) \tag{11}$$

Ignoring the higher order terms by assuming f to be quadratic around x_0 and solving for the minimum of x by setting the lefthand side of Equation 10 to 0, we obtain Newton's method:

$$x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1} \nabla f(x_i) \tag{12}$$

where x_0 is replaced by x_i and x by x_{i+1}.

LMNN is designed to approach second-order training speed without having to compute the Hessian matrix, which is

$$\mathbf{H} = \mathbf{J}^T \mathbf{J} \tag{13}$$

The gradient can be computed as

$$\mathbf{g} = \mathbf{J}^T \mathbf{e} \tag{14}$$

where $\mathbf{J}$ is the Jacobian matrix that contains the first derivatives of the network errors with respect to the weights and biases, and $\mathbf{e}$ is the vector of network errors. Levenberg proposed an algorithm that combines the above equations:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - [\mathbf{H} + \lambda \mathbf{I}]^{-1} \mathbf{g} \tag{15}$$

Levenberg's algorithm can be outlined as follows:

1. Update weights as in Equation 15;
2. Evaluate the error of the new parameter vector;
3. If the error has increased then reset the weights to their previous values and increase λ by a significant factor and goto 1;
4. If the error had decreased then accept the new values for the weights and decrease λ by a factor a significant factor and goto 1.

A problem with the above algorithm is that when λ is large, the Hessian matrix is not used. Marquardt improved the algorithm by replacing the identity matrix in Equation 15 with the diagonal of the Hessian, resulting the Levenberg-Marquardt update rule

$$\mathbf{x}_{i+1} = \mathbf{x}_i - (\mathbf{H} + \lambda diag[\mathbf{H}])^{-1} \mathbf{g} \tag{16}$$

By using singular value decomposition (SVD) and other techniques to compute the inverse, the cost of the updates is far less than the costs involved in computing the gradient descent for parameters numbering only in the hundreds, but eventually the costs become prohibitive when weight size increases into the thousands.

2.2 Classifier Ensembles

This section provides a description of the aggregating methods used in our experiments, viz., Bagging (BA), Class Switching (CW), and Random Subspace (RS).

2.2.1 Bagging (BA)

The term *bagging* was first introduced in [10] as an acronym of Bootstrap AGGregatING. The idea is to generate random bootstrap training subsets, with replacement, from the master training set. Base classifiers are then trained on each subset and the results combined using a majority vote rule. That is, the combiner evaluates the testing samples by querying each of the base classifiers on the sample and then outputting the majority opinion.

Breiman's [10] two main arguments for Bagging effectiveness are the following: 1) running many trials on uniform samples of a population results in less variant statistical results, and 2) the majority opinion reduces noise-induced errors made by a small minority of the base classifiers.

Key to the success of using this algorithm is the selection of appropriate base classifiers. To guarantee diversity, the classifiers should be unstable, i.e., small variations in the training set should produce large changes on the classifiers, otherwise the ensemble will not outperform the individual classifiers. Typical unstable classifiers are neural networks, decision trees, regression trees, and linear regression.

Outline of Bagging

A bootstrap subset is generated, with replacement, by uniformly sampling m instances from the master training set. T bootstrap subsets $B_1, B_2, ..., B_T$ are generated and a classifier C_i is built from each bootstrap subset, B_i. The combiner, C^*, is built from $C_1, C_2, ..., C_T$ whose output is the class predicted most often by the classifiers, with ties broken arbitrarily.

Below is a simple pseudocode description of the bagging algorithm.

In the training process

```
For t = 1,2, ... T do:
    1. Build a random subset $\mathbf{x}_t$ taking randomly selected samples from the original
       training set;
    2. Train a classifier $C_t$ using the subset $\mathbf{x}_t$;
    3. Add the classifier to the ensemble.
```

In the testing process

Build the final decision rule by combining the results of the classifiers. Several decision rule can be used to combine the classifier, e.g., majority voting, the sum rule, and the max rule [40].

2.2.2 Class Switching (CW)

Class Switching is an ensemble method based on the creation of new training sets obtained by changing the class labels of the training patterns. The class label of each example is switched according to a probability function that depends on an overall switching rate. Thus, in each new training set, the label of a fixed fraction, p, of the training patterns of the master training set is selected at random for switching.

Outline of Class Switching

As outlined in [54], the class switching ensemble method is characterized by the following rule:

$$\mathrm{P}_{ji} = p/(K \quad - 1), \mathrm{i} \neq \mathrm{j} \,,$$

$$\mathrm{P}_{ii} = 1 - p$$

where: p is the switching rate, P_{ji} is the probability that an example with label i gets the label j, P_i is the proportion of elements of class i, and K is the number of classes.

2.2.3 Random Subspace (RS)

Random subspace is a method for reducing dimensionality by randomly sampling subsets of features and improving accuracy by aggregating the resulting base classifiers. The seminal paper on this method is [34].

Outline of Random Subspace

As outlined in [34], the Random Subspace ensemble method is characterized by three steps:

1. Given a d-dimensional data set $\mathscr{D} = \{(x_j,t_j)|1 \leq j \leq m\}, x_j \in \mathrm{X} \subset \mathrm{R}^d, t_j \in \mathrm{C} = \{1,...,c\}$, n new projected k-dimensional data sets D_i=$\{(P_i(x_j),t_j)|1 \leq j \leq m\}$ are generated $(1 \leq i \leq n)$, where P_i is a random projection $P_i : \mathrm{R}^d \rightarrow \mathrm{R}^k$. P_i is obtained by random selecting, through the uniform probability distribution, a k-subset $A = \{\alpha_1,...,\alpha_k\}$ from $\{1,2,...,d\}$ and setting $P_i(x_i,...,x_d) = (x_{\alpha 1},...,x_{\alpha k})$;
2. Each new data set D_i is given in input to a fixed learning algorithm L which outputs the classifiers h_i for all $i, 1 \leq 1 \leq n$;
3. The final classifier h is obtained by aggregating the base classifiers $h_i,...,h_n$ through a given decision rule.

2.3 Resources

Because of its excellent visualization tools and platform independence, MATLAB [76] by MathWorks is commonly used for experimenting with face classification algorithms. Numerous toolboxes have been developed that provide MATLAB users with routines for handling images and statistical pattern recognition tasks. MathWorks, for instance, offers an excellent image processing toolbox. MathWorks also produces a neural network toolbox for designing and visualizing neural network algorithms, with built-in support for many common neural network algorithms, such as LMNN.

An excellent MATLAB toolbox for experimenting with statistical pattern recognition is PRTools [83]. It is free for academic research. PRTools provides over 200 routines, including PCA, LMNN, Oja's subspace, and SVM. The SVM implementation in PRTools, however, is limited. For a more comprehensive package, the OSU SVM MATLAB toolbox developed by Ohio State University is an excellent choice. It is available at http://sourceforge.net. Links to additional software and resources are available at http://www.face-rec.org.

3 Study Design

In the experiments presented in this chapter, single classifiers and classifier ensembles are trained to detect the social meanings people perceive in facial morphology. Our first concern in designing our study was developing a sound ground truth for this problem domain. Our goal was to collect a set of faces that exhibit strong human consensus in a comprehensive set of trait categories. The traits selected for this study are a modification of Rosenberg's [25, 29, 71] meta-analysis of a broad range of categories and include the following: dominance, intelligence, maturity, sociality, trustworthiness, and warmth.

As noted in the introduction, our task is unusual in that our ground truth is not based on factual information about the subjects' faces. In most face recognition and classification applications, faces are associated with classes that are derived either from the subject's self-report (age, gender, and emotional state) or from different views, spatial as well as temporal, of the same person. The division of faces into relevant classes poses few problems, as the classes are clearly definable. In the classification task of matching human impressions of faces, however, the division of faces into relevant trait classes is not a straightforward process. It is complicated by the fact that most faces fail to elicit strong opinions in any given trait dimension and by the fact that human beings, while consistent in their ratings, are not in total agreement.

Figure 6 outlines the steps we took to develop our database of faces. In Section 3.1, we describe the process we used to generate 480 stimulus faces. We also discuss some of the issues involved in selecting different facial representations (e.g., artificially constructed faces, 2-D photographs, and 3-D scans) and justify our decision to artificially construct faces from the popular composite program FACES [30]. In

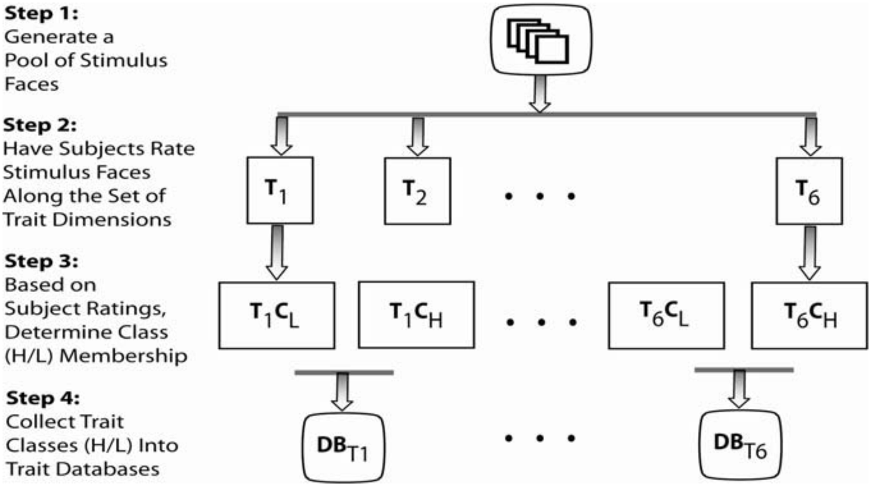

Step 1:
Generate a Pool of Stimulus Faces
Step 2:
Have Subjects Rate Stimulus Faces Along the Set of Trait Dimensions
Step 3:
Based on Subject Ratings, Determine Class (H/L) Membership
Step 4:
Collect Trait Classes (H/L) Into Trait Databases
T1
T2
. . .
T6
T1CL
T1CH
. . .
T6CL
T6CH
DBT1
. . .
DBT6

Fig. 6 Steps taken to generate a database of representative faces for each trait dimension (T_{1-6}) with two classes, high (C_H) and low (C_L)

Section 3.2, we describe our experimental design for assessing subject ratings of the stimulus faces along the six trait dimensions using a 3-point scale. In Section 3.3, we describe the process we used in steps 3 and 4 for determining face membership into the two trait classes of *high* and *low* for each trait dimension. It should be noted here that a stimulus face can belong to more than one trait database. Faces can be rated as both submissive and trustworthy, for instance. However, within any trait database, a stimulus face can only belong to one of the two trait classes. A face, for example, cannot be both high and low in perceived dominance.

3.1 Step 1: Generation of Stimulus Faces

In attribution studies in the person perception literature, stimulus faces are typically of four types: photographs of faces, drawings of faces, faces pieced together using facial composite products such as Identi-Kit [12], and faces that have been altered using a variety of geometrical transformations (see for instance [77], where a cardioidal transform is used to vary facial maturity).

Each of these facial representations offers some attractive benefits. Two-dimensional photographs and facial composites have been widely studied in the person perception literature, and compared to 3-D scans are easier to collect. An advantage using facial composite programs, whether two-dimensional or three-dimensional, is that the contributions of individual facial features in the attribution process can more easily be investigated.

Each facial representation is also problematical. A danger in using a dataset of faces that have been *artificially* produced is that the classifier systems might not generalize well enough to handle photographic images of actual faces. It might be thought that using photographs of actual faces would address this potential problem. However, photographs are two-dimensional representations, and it could be argued that people form impressions of faces *in situ* based on multidimensional views. Three dimensional scans of actual faces also present representational dilemmas. How faces are seen in space for instance could affect viewer ratings. Will the viewer control how the scans are viewed or will the scanned faces move on their own? Even judging films of faces is problematical, as the perspective of the camera is typically artificial and stationary.

In psychological studies, large databases of faces are not required. It is not uncommon for subjects to evaluate fewer than twenty faces. In building classification systems, larger numbers of samples are necessary. Furthermore, the faces need to strongly represent the various trait classes. Most faces are not extreme in their attributions, so finding large numbers of faces that distinctly represent various traits is difficult.

To generate as many representative faces as possible, we decided that it would be best to construct faces artificially. We asked four college art students (all female seniors) to generate 120 faces (60 female and 60 male) using the program FACES. This produced a total of 480 stimulus faces. The artists were given three months to complete the task and were asked to generate even groups of faces (5 male and 5 female) that they thought would be perceived by others as intelligent, unintelligent, mature, immature, warm, cold, social, unsocial, dominant, submissive, trustworthy, and untrustworthy. The artists were given the same definitions of these terms as were later given the subjects who rated the stimulus faces using these same trait descriptors (see Appendix for these definitions). Thus, we hoped to obtain at least 40 faces in each trait class that would be verified by subjects to produce the impressions intended.

The artists were also instructed to use as many different facial features in the program's database as possible, with the caveat that they make the emotional expressions of the faces as neutral as possible. The program FACES contains a fairly large set of individual features: 512 eyes, 541 noses, 570 lips, 423 jaws, 480 eyebrows, and 63 foreheads, to list some of the more important features, so the faces were generally unique in appearance, as illustrated in Figure 7.

3.2 Step 2: Assessing Trait Impressions of Stimulus Faces

Once the stimulus faces were generated, they were evaluated in step 2 by 20 human subjects as detailed below.

Subjects. A total of 80 students, recruited from the same university as the artists, were asked to judge the stimulus faces. An equal number of male and female subjects participated in the study. A total of 62 were undergraduate and 18 were graduate. The average age was 24 and ranged from 18 to 61. Each student received extra credit in a Computer Information Systems course for participating in the study. The graduate students were in the college of business and were working towards MBA and MHA degrees. The undergraduates were mostly business and science majors.

The majority of students were white (69), followed by African (5), Asian (3), Hispanic (2) and other (1).

Dependent Measures. Each subject judged a full set of 120 faces created by one of the four artists along the six trait dimensions using a forced 3-point scale with associated descriptors. Thus, each image was judged by 20 subjects. The 1 and 3 values were given the bipolar trait descriptors of dominant/submissive, intelligent/unintelligent, mature/immature, social/unsocial, trustworthy/untrustworthy, and warm/cold, and their positions were randomized for each trait dimension and for each image. *Neutral* was always located at the middle 2 value.

Subjects were given access to the trait definitions and, in some cases, behavioral potential questions modeled after Berry and Brownlow [7] and Zebrowitz and Motepare [88]. The Appendix provides the description of the traits and the behavioral potential questions that were given to both the artists and the subjects.

Apparatus. The program that administered the survey was located on a campus server and was made available to the participants in the computer labs located across campus. Due to the large number of faces each subject was asked to rate, they were given one week to complete the task.

3.3 Step 3: Division of Stimulus Faces into Trait Class Sets

A complete analysis of human variance is not presented in this study, as the objective was not to perform another facial attribution study. This is a topic that has been well researched, and people of different ages, genders, races, and cultures have been shown to be remarkably consistent in their judgments [2, 90]. Rather the study design was geared solely towards obtaining human judgments of the 460 faces in order to extract faces that unambiguously elicit specific impressions. Table 2 presents the mean ratings and standard deviations of the 460 faces for each of the six trait dimensions. Standard deviations were less than 1.0, and the average rating of the faces in each dimension was close to neutral (2.0).

Membership in each trait class of high and low required that a face met the following specifications:

1. That the standard deviation in the ratings was less than 1.0;

Table 2 Rater Means and Standard Deviations Per Trait of the 460 Stimulus Faces

Trait Dimension Descriptor	*Trait Dimension Means*	*Standard Deviations*
Intelligence	2.20	0.794
Dominance	2.00	0.782
Maturity	2.10	0.822
Sociality	1.99	0.784
Trustworthiness	2.07	0.771
Warmth	2.00	0.728

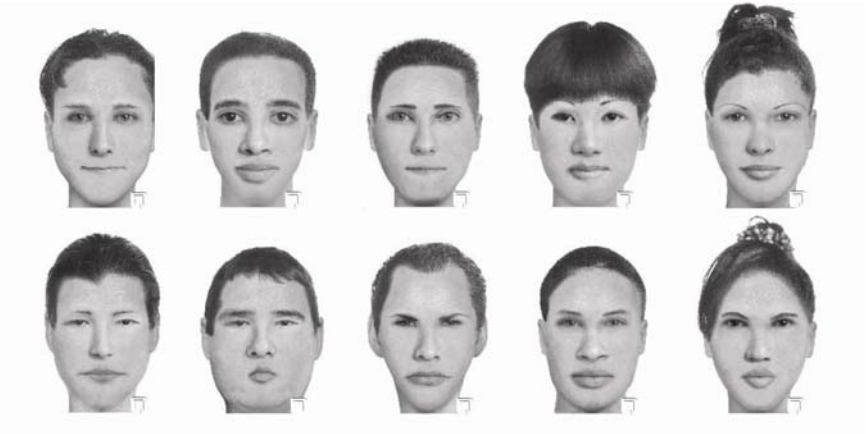


Fig. 7 Samples from the two classes (high and low) of warmth. The top row faces were rated significantly higher in warmth. The bottom row faces were rated significantly lower in warmth

Table 3 Number of Images in the Two Classes (High and Low) of Each Trait Dimension

Trait Dimension	***Attribution Class***	***Number***
Intelligence	Low High	39 140
Dominance	Low High	99 102
Maturity	Low High	49 126
Sociality	Low High	126 117
Trustworthiness	Low High	97 151
Warmth	Low High	139 146

2. That the mean rating was less than 1.6 for low membership and greater than 2.4 for high membership;
3. That the mode matched the class (1 for low and 3 for high)

To illustrate class membership, in Figure 7, a sample of faces that fell into the high and low classes of warmth are presented.

Table 3 lists the total number of images that fell into the two classes for each trait dimension. The average number of images in each class is 111 (minimum 39 and maximum 151). Except for low intelligence (39), the number of images that fell into each class greatly exceeded our minimum expectation of 40 faces. Since trait dimensions are correlated (recall in the introduction how morphological features of the overgeneralization effects were associated with clusters of traits), the total number of images is greater than 460 because many images produced significant trait impressions in more than one dimension.

4 Classification Experiments

In this section we describe our classification experiments using the six trait databases. In Section 4.1 we describe the basic system architecture, and in Section 4.2 we present our experimental results.

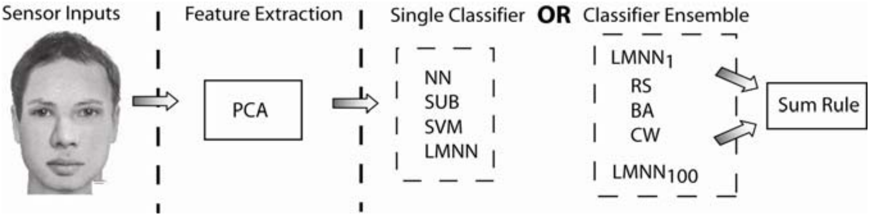


Fig. 8 Basic System Architecture

4.1 System Architecture

Figure 8 provides a basic schematic of our system architecture. First, we transform the raw sensor inputs (gray scale values) using PCA, where the preserved variance is 0.98. Then we train both single classifier systems (NN, SUB, SVM, and LMNN) and different ensembles of 100 LMNNs, using RS, BA, and CW.

Our single classifier systems included the following classifiers[1]:

- PCA+NN, classifier based on the PCA coefficients and 1-nearest neighbor classification using the Euclidean distance;
- PCA+SUB, classifier based on the PCA coefficients and Oja's subspace maps, where subspaces are computed for each class of the dataset and the distance of the test patterns to these subspaces are used to classify the data (p=(3/5)×[($K-$ 1)/K]);
- PCA+SVM, classifier based on the PCA coefficients and a Linear Support Vector Machine [84];
- PCA+ LMNN, classifier based on the PCA coefficients and a Levenberg-Marquardt neural net [24], using 5 hidden nodes.

As noted in Section 2, collaborative methods, such as RS, BA, and CW, work best if unstable classifiers are used in building the ensembles [46]. For our ensemble experiments, we selected the Levenberg-Marquardt neural net (LMNN) with 5 hidden nodes for comparison purposes. The ensembles were constructed using the following three methods (see Section 2 for a more detailed description of each method):

[1] Implemented as in PrTools 3.1.7 ftp://ftp.ph.tn.tudelft.nl/pub/bob/prtools/prtools3.1.7

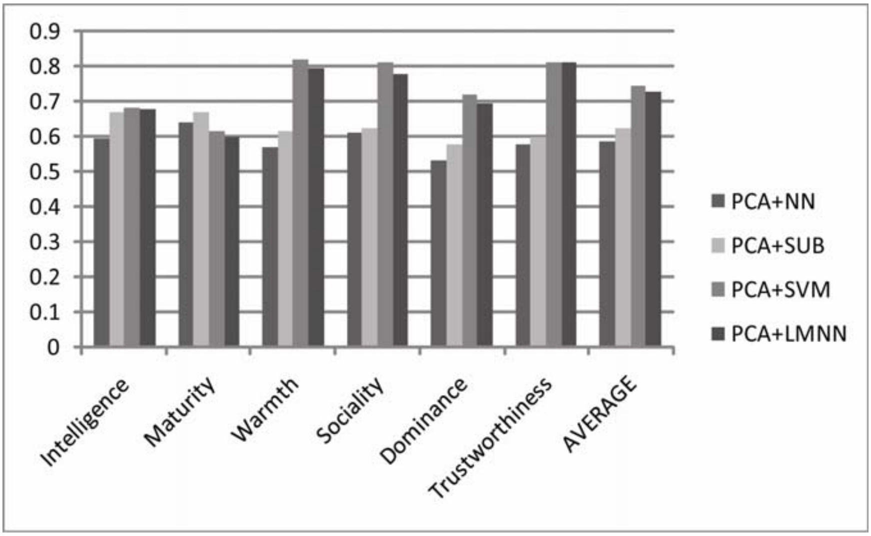


Fig. 9 Comparison of single classifier performance (average AROC obtained in 20 runs) for the six trait dimensions

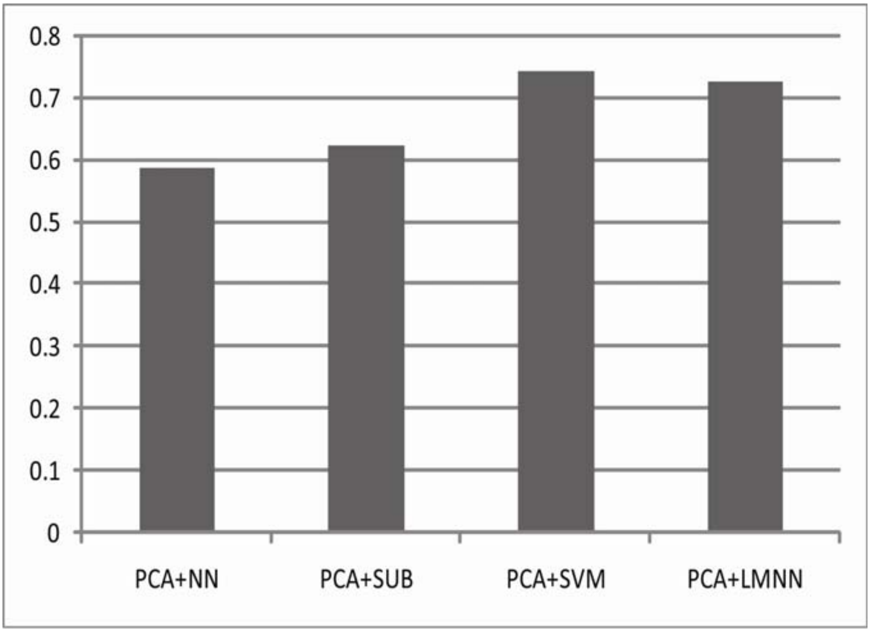


Fig. 10 Comparison of single classifier performance averaged across all six trait dimensions

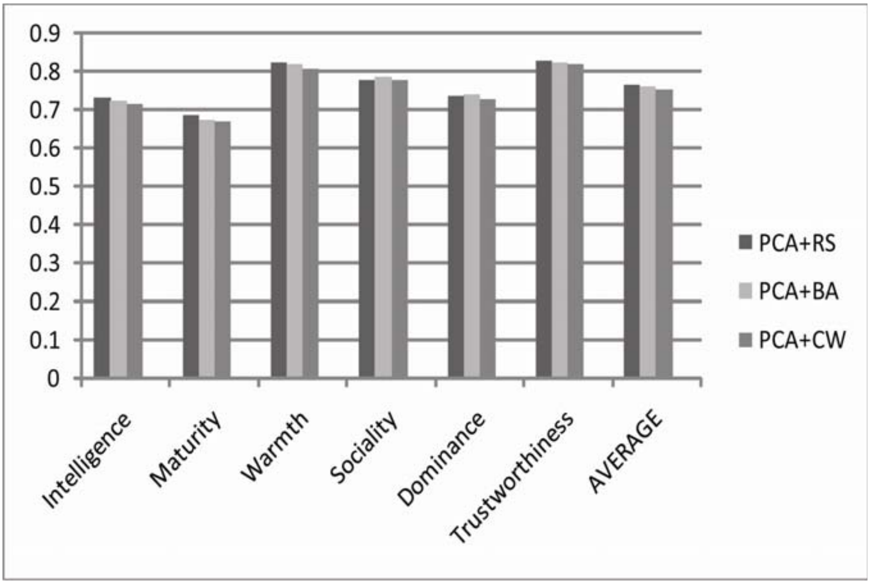


Fig. 11 Comparison of ensemble performance (average AROC obtained in 20 runs) for the six trait dimensions

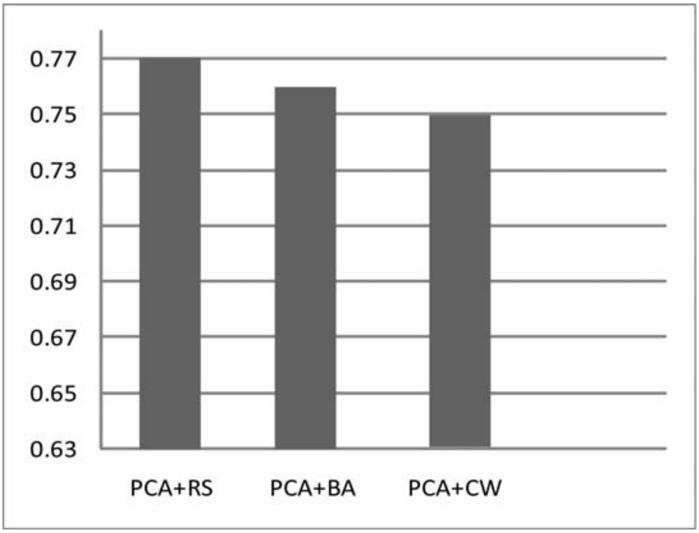


Fig. 12 Comparison of ensemble classifier performance averaged across all six trait dimensions

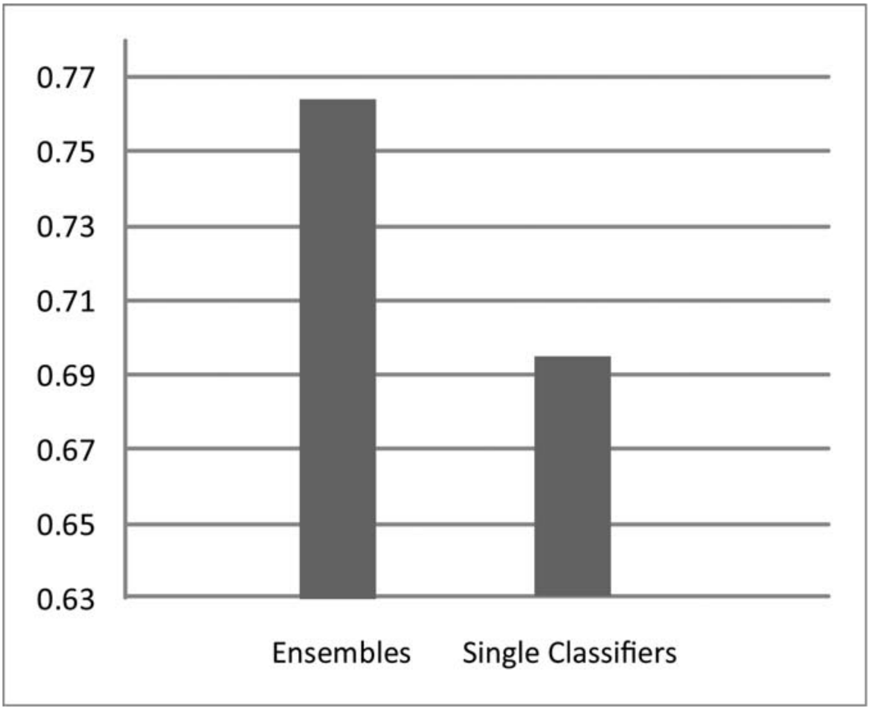


Fig. 13 Comparison of ensemble and single classifier performance averaged across all six trait dimensions

- PCA+RS, classifier based on the PCA coefficients and a RS ensemble of 100 LMNNs. Recall from Section 2 that in RS the individual classifiers use only a subset of all features for training and testing. The percentage of the features retained in each training set was set to 50%;
- PCA+BA, classifier based on the PCA coefficients and a BA ensemble of 100 LMNN. Given a training set S, BA generates K new training sets $S_1, \ldots, S_K$; each new set S_i is used to train exactly one classifier. Hence an ensemble of individual classifiers is obtained from K new training sets;
- PCA+CW, classifier based on the PCA coefficients and a CW ensemble of 100 LMNN. As explained more fully in Section 2, CW creates an ensemble that combines the decisions of classifiers generated by using perturbed versions of the training set where the classes of the training examples are randomly switched.

4.2 Results

The performance indicator adopted in this work is the area under the Receiver Operating Characteristic curve (AROC) [51]. As explained in Section 2, AROC is a two-dimensional measure of classification performance that plots the probability of

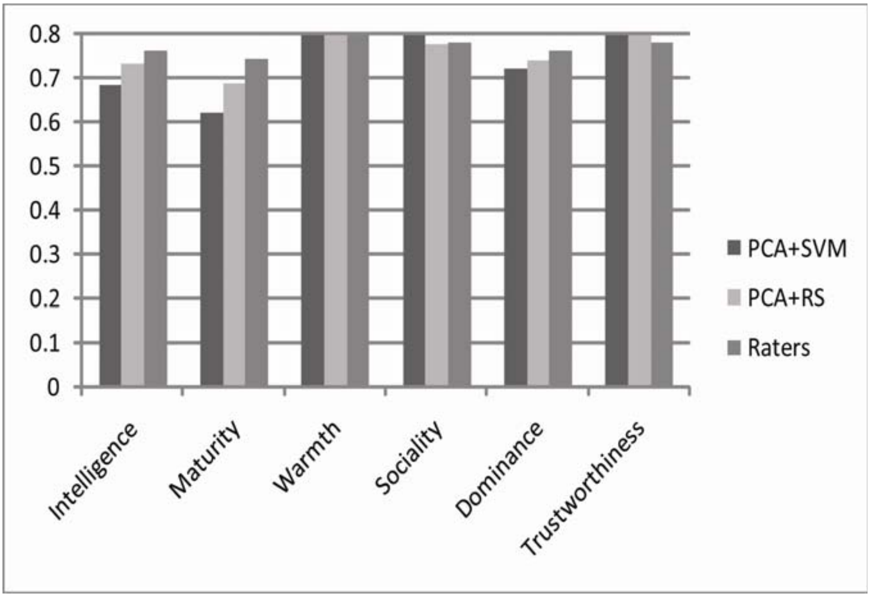


Fig. 14 Comparison of best single (PCA+SVM) and ensemble (PCA+RS) performance (average AROC obtained in 20 runs) with subject (Raters) performance (average AROC) for the 6 trait dimensions

classifying the genuine examples correctly against the rate of incorrectly classifying impostor examples.

In Figure 9, we compare the performance of single classifiers on each of the six trait dimensions. SVM and LMNN greatly outperformed NN and SUB on the trait dimensions of warmth, sociality, dominance, and trustworthiness. Looking at Table 4 and Figure 10, we find that the average performance of SVM and LMNN are very close, with both SVM and LMNN performing best across all six trait dimensions. However, referring back to Figure 9, we see that SVM and LMNN performed relatively poorly, compared with SUB and NN, on the trait dimension of maturity. None of the single classifiers in our experiments was able to perform well on all six dimensions.

In Figures 11 and 12, we compare the average AROC of the ensemble experiments using the six databases. As seen in Table 4, RS, BA, and CW perform comparatively well, with RS performing slightly better than BA and CW. As with the single classifier systems, ensembles had more difficulty classifying faces perceived as intelligent and mature.

Looking at Table 4 and Figure 13, we can compare results between the single classifier systems and the ensembles. The ensembles clearly outperform NN and SUB. SVM and LMNN are close in performance to the ensembles; however, as we can see in Table 4, the ensembles, unlike SVM and LMNN, performed comparatively well across all six trait dimensions.

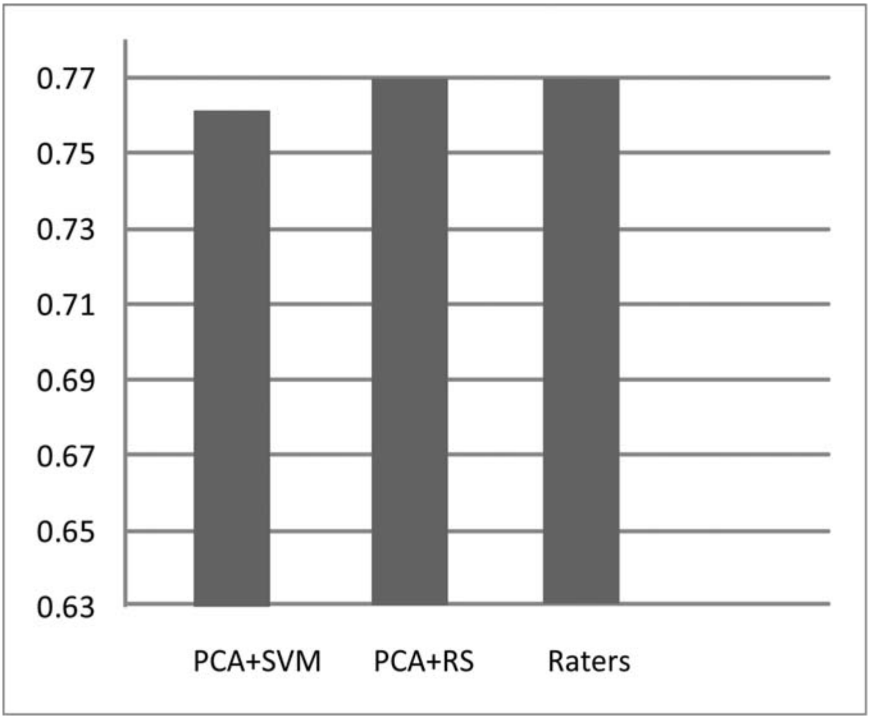


Fig. 15 Comparison of best ensemble classification (PCA+RS) and best single classifier (PCA+SVM) performance with subject (Raters) performance averaged across all six trait dimensions

Table 4 Performance Comparison (Average AROC Obtained in 20 Runs) Between Single Classifier Systems, Ensembles, and Human Subjects (Raters) for the Six Trait Dimensions

Classifier	**Intelligence**	**Maturity**	**Warmth**	**Sociality**	**Dominance**	**Trustworthiness**	**Average**
PCA+SVM	0.68	0.62	0.82	0.81	0.72	0.81	0.74
PCA+NN	0.59	0.64	0.57	0.61	0.53	0.58	0.59
PCA+SUB	0.67	0.67	0.61	0.62	0.58	0.60	0.63
PCA+LMNN	0.68	0.60	0.80	0.78	0.62	0.81	0.73
PCA+RS	0.73	0.69	0.82	0.78	0.74	0.83	0.77
PCA+BA	0.73	0.67	0.82	0.79	0.74	0.83	0.76
PCA+CW	0.71	0.67	0.81	0.78	0.73	0.82	0.75
Raters	0.76	0.75	0.80	0.78	0.76	0.78	0.77

Looking at Table 4, where we report the average AROC obtained by the subjects, and Figures 14 and 15, we can see that the performance obtained by ensembles is similar to the performance of the average rater, with RS exactly matching the raters in the averaged performance of all six trait dimensions. This result leads us to believe that machines are as capable of classifying faces according to the impressions they make on the general observer as are most human beings.

5 Conclusion

In this chapter we present unique face classification experiments using a variety of collaborative methods. The experiments are unique because the systems were not asked to classify faces according to such factual information as identity and gender but rather the systems had to match the human perception of faces in terms of the social impressions they make on the average observer.

One contribution of the study reported in this chapter was the development of a sound ground truth for this problem domain. Our goal was to collect a set of faces that exhibit strong human consensus in a comprehensive set of trait categories. To accomplish this objective, four artists were asked to construct 480 stimulus faces, using the composite program FACES, with an eye towards making faces they thought were clearly intelligent, unintelligent, mature, immature, warm, cold, social, unsocial, dominant, submissive, trustworthy, and untrustworthy. Subjects then rated the 480 faces using the same twelve descriptors. Since traits are correlated, this process succeeded in creating trait classes that averaged over one hundred faces each, a vast improvement over the databases of faces we used in earlier work (see [9]).

Single classifiers and ensembles were then trained to match the bipolar extremes of the faces in each of the six trait dimensions of intelligence, maturity, warmth, sociality, dominance, and trustworthiness. With performance measured by AROC and averaged across all six dimensions, results show that single classifiers, especially linear SVMs (0.74) and LMNN (0.73), performed as well as human raters (0.77). These single classifiers, however, performed poorly in the trait dimension of maturity. Ensembles of 100 LMNNs, constructed using BA (0.76), SUB (0.77), and CW (0.75), compared equally well to rater performance, but were better than the single classifiers at handling all six trait dimensions. The Random subspace AROC, averaged across the six dimensions, exactly matched rater performance. We concluded that machine learning methods, especially ensembles, are as capable of *perceiving* the social impressions faces make on the general observer as are most human beings.

Although research shows that people perceive personality even in abstract drawings of faces [15], one shortcoming in developing the system reported in this chapter is the possibility that classifiers trained on artificially constructed faces will not generalize to natural faces. We are currently developing studies to determine how well our ensembles, trained with this dataset, are able to match human raters of photographs of people. In addition, we are developing a dataset of photographs of faces that have equally large numbers of faces in each of the twelve trait classes.

As noted in the introduction, developing models of the human perception of the social meanings of faces may have value in a number of fields, including social psychology and human-computer interaction. Certainly, the human-like interfaces and robots of the future will need to be able to see faces and other objects as human beings see them, if they are to have more than a superficial social engagement with us. It is not enough in human society simply to recognize what an object is; one must also be aware of the cultural layers of meanings that envelop each object. Our experiments demonstrate that it is possible for machines to match some of these cultural meanings to attributes possessed by the objects. For socially interactive interfaces and robots to be believable, however, they will need the ability to integrate a host of social impressions. Building machines that perceive the social meanings of objects will involve further research in the exciting area of collaborative computational intelligence.

Appendix

Table 5 Definition of each trait dimension and behavioral potential questions (modelled after Berry and Brownlow [7] and Zebrowitz and Motepare [88]) as given to the subjects who evaluated the stimulus faces

Dominant, Submissive, Neutral	Here we are looking at how dominating the person looks. **Dominant:** Is a person who is most likely to tell other people what to do. **Submissive:** Is a person who usually follows others and is not very assertive. A helpful question might be: "Does s/he look like someone who would be the kind of roommate who would comply with most of your wishes about the furniture arrangement, quiet hours, and house rules?"
Intelligent, Unintelligent, Neutral	Here we are looking at how intelligent the person looks. **Intelligent:** Is a person who is possible very educated, capable, and interested in intellectual work. **Unintelligent:** Is a person who probably does not value school as s/he was not good at school subjects. A helpful question might be: "Does s/he look like someone you would learn from when discussing such topics as art, politics, philosophy, or science?"
Mature, Immature, Neutral	Here we are looking at how responsible the person looks. **Mature:** Is a person who acts like an adult and behaves responsibly. **Immature:** Is a person who behaves in a childish or irresponsible manner. A helpful question might be: "Does s/he look like someone you could trust to take on important responsibilities?"
Trustworthy, Untrustworthy, Neutral	Here we are looking at how honest the person looks. **Trustworthy:** Is a person who is mostly honest and is not likely to steal, lie, or cheat. **Untrustworthy:** Is a person who is often not honest and who possible steals, lies, or cheats. A helpful question might be: "Does s/he look like someone you would ask to watch your backpack while you made a visit to the restroom?"
Social, Unsocial, Neutral	Here we are looking at how social the person looks. **Social:** Is a person who is most likely very outgoing, extroverted, and who enjoys parties and other social activities. **Unsocial:** Is a person who is introverted, a loner, shy, and who would prefer to stay home rather than go out. A helpful question might be: "Does s/he look like someone you would invite to a party to enliven it?"
Cold, Warm, Neutral	Here we are looking at how approachable the person is. A helpful question might be: "Does s/he look like someone who would turn a cold shoulder to your attempts at friendly conversation?"

References

1. Abate, A.F., Nappi, M., Riccioa, D., Sabatinoa, G.: 2D and 3D face recognition: A survey. Pattern Recognit. Lett. 14, 1885–1906 (2007)
2. Albright, L., Malloy, T.E., Dong, Q., Kenny, D.A., Fang, X.: Cross-cultural consensus in personality judgments. J. Personal and Soc. Psychol. 3, 558–569 (1997)
3. Alcock, D., Solano, J., Kayson, W.A.: How individuals' responses and attractiveness influence aggression. Psychol. Rep. 3(2), 1435–1438 (1998)
4. Alpaydin, E.: Introduction to machine learning. MIT Press, Cambridge (2004)
5. Altınçay, H., Demirekler, M.: An information theoretic framework for weight estimation in the combination of probabilistic classifiers for speaker identification. Speech Commun. 4, 255–272 (2000)
6. Bellman, R.: Adaptive control process: A guided tour. Princeton University Press, Princeton (1961)
7. Berry, D.S., Brownlow, S.: Were the physiognomists right? Personal and Soc. Psychol. Bull. 2, 266–279 (1989)
8. Berry, D.S., McArthur, L.Z.: Perceiving character in faces: The impact of age-related craniofacial changes on social perception. Psychol. Bull. 1, 3–18 (1986)
9. Brahnam, S.: Modeling physical personalities for virtual agents by modeling trait impressions of the face: A neural network analysis. The Graduate Center of the City of New York, Department of Computer Science, New York (2002)
10. Breiman, L.: Bagging predictors. Mach. Learn. 2, 123–140 (1996)
11. Breiman, L.: Random forest. Mach. Learn. 1, 5–32 (2001)
12. Bruce, V.: Recognising faces. Lawrence Erlbaum Associates Publishers, London (1988)
13. Brunelli, R., Poggio, T.: Hyperbf networks for gender classification. In: DARPA Image Understanding Workshop, pp. 311–314 (1992)
14. Brunelli, R., Poggio, T.: Face recognition: Features versus templates. IEEE Trans. Pattern Anal. and Mach. Intell. 10, 1042–1052 (1993)
15. Brunswik, E.: Perception and the representative design of psychological experiments. University of California Press, Berkeley (1947)
16. Bull, R., Rumsey, N.: The social psychology of facial appearance. Springer, Heidelberg (1988)
17. Burton, A.M., Bruce, V., Dench, N.: What's the difference between men and women? Evidence from facial measurement. Percept. 2, 153–176 (1993)
18. Chellappa, R., Wilson, C.L., Sirohey, S.: Human and machine recognition of faces: A survey. Proceedings of the IEEE, pp. 705–740 (1995)
19. Cottrell, G.W., Fleming, M.K.: Face recognition using unsupervised feature extraction. In: International Conference on Neural Networks, pp. 322–325 (1990)
20. Cottrell, G.W., Metcalfe, J.: EMPATH: Face, emotion, and gender recognition using holons. In: Touretzky, D. (ed.) Adv. Neural Inf. Process Syst., pp. 564–571. Morgan & Kaufman, San Mateo (1991)
21. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classificiation. IEEE Trans. Inf. Theory 1, 21–27 (1967)
22. Cristianini, N., Shawe-Taylor, J.: An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge (2000)
23. Czyz, J., Kittler, J., Vandendorpe, L.: Multiple classifier combination for face-based identity verification. Pattern Recognit. 7, 1459–1469 (2004)
24. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley, New York (2000)

25. Eagly, A.H., Ashmore, R.D., Makhijan, M.G., Longo, L.C.: What is beautiful is good, but ...: A meta-analytic review of research on the physical attractiveness stereotype. Psychol. Bull. 1, 109–128 (1991)
26. Edelman, B.E., Valentin, D., Abdi, H.: Sex classification of face areas: How well can a linear neural network predict human performance. J. Biol. Syst. 3, 241–264 (1998)
27. Efron, B.: The jackknife, the bootstrap and other resampling plans. SIAM, Philadelphia (1982)
28. Enlow, D.H., Hans, M.G.: Essentials of facial growth. W. B. Saunders Company, Philadelphia (1996)
29. Feingold, A.: Good-looking people are not what we think. Psychol. Bull. 2, 304–341 (1992)
30. Freierman, S.: Constructing a real-life mr. potato head. Faces: The ultimate composite picture. The New York Times D:6 (2000)
31. Golumb, B.A., Lawrence, D.T., Sejnowshi, T.J.: Sexnet: A neural network identifies sex from human faces. Adv. Neural Inf. Process Syst., 572–577 (1991)
32. Guo, G., Li, S.Z., Chan, K.L.: Support vector machines for face recognition. Image and Vis. Comput., 631–638 (2001)
33. Heisele, B., Ho, P., Poggio, T.: Face recognition with support vector machines: Global versus component-based approach. In: The Eighth IEEE International Conference on Computer Vision, Vancouver, BC, pp. 688–694 (2001)
34. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. and Mach. Intell. 8, 832–844 (1998)
35. Jain, A., Huang, J.: Integrating independent components and linear discriminant analysis for gender classification. In: Sixth IEEE International Conference on Automatic Face and Gesture Recognition, pp. 159–163 (2004)
36. Jain, A.K., Dubes, R.C., Chen, C.C.: Bootstrap techniques for error estimation. IEEE Trans. Pattern Anal. and Mach. Intell. 5, 628–633 (1987)
37. Jain, A.K., Duin, R.P.W., Mao, J.: Statistical pattern recognition: a review. IEEE Trans. Pattern Anal. and Mach. Intell. 1, 4–37 (2000)
38. Kanghae, S., Sornil, O.: Face recognition using facial attractiveness. In: The 2nd International Conference on Advances in Information Technology, Bangkok, Thailand (2007)
39. Keating, C.F., Mazur, A., Segall, M.H.: A cross-cultural exploration of physiognomic traits of dominance and happiness. Ethol. and Sociobiol., 41–48 (1981)
40. Kittler, J.: On combining classifiers. IEEE Trans. Pattern Anal. and Mach. Intell. 3, 226–239 (1998)
41. Kohonen, T.: Associative memory: A system theoretic approach. Springler, Berlin (1977)
42. Kosugi, M.: Human-face search and location in a scene by multi-pyramid architecture for personal identification. Syst. and Comput. Jpn. 6, 27–38 (1995)
43. Kuncheva, L.I.: Clustering-and-selection model for classier combination. In: Knowledge-Based Intelligent Engineering Systems and Allied Technologies, Brighton, pp. 185–188 (2000)
44. Kuncheva, L.I.: Combining pattern classifiers: Methods and algorithms. Wiley, New York (2004)
45. Kuncheva, L.I.: Diversity in multiple classifier systems. Inf. Fusion 1, 3–4 (2005)
46. Kuncheva, L.I., Whitaker, C.J.: Measures of Diversity in Classifier Ensembles and their Relationship with the ensemble accuracy. Mach. Learn. 2, 181–207 (2003)
47. Langlois, J.H., Kalakanis, L., Rubenstein, A.J., Larson, A., Hallam, M., Smoot, M.: Maxims or myths of beauty? A meta-analytic and theoretical review. Psychol. Bull. 3, 390–423 (2000)

48. Lanitis, A., Taylor, C.J., Cootes, T.F.: Automatic interpretation and coding of face images using flexible models. IEEE Trans. Pattern Anal. and Mach. Intell. 7, 743–756 (1997)
49. Lanitis, A., Taylor, C.J., Cootes, T.F.: Toward Automatic Simulation of Aging Effects on Face Images. IEEE Trans. Pattern Anal. and Mach. Intell. 4, 442–455 (2002)
50. Levenberg, K.: A Method for the solution of certain nonlinear problems in least squares. Q Appl. Math. 2, 164–168 (1944)
51. Ling, C.X., Huang, J., Zhang, H.: Auc: A better measure than accuracy in comparing learning algorithms. In: Canadian Conference on Artificial Intelligence 2003, Halifax, Canada, pp. 329–341 (2003)
52. Lu, X., Jain, A.K.: Ethnicity identification from face images. In: SPIE: Biometric Technology for Human Identification Conference: Biometric Technology for Human Identification, Orlando, FL, pp. 114–123 (2004)
53. Marquardt, D.: An algorithm for least-squares estimation of nonlinear parameters. SIAM J. Appl. Math., 431–441 (1963)
54. Martínez-Muñoz, G., Suárez, A.: Switching class labels to generate classification ensembles. Pattern Recognit. 10, 1483–1494 (2005)
55. Martinez, A.M., Benavente, R.: The ar face database. CVC Technical Report #24 (1998), http://rvl1.ecn.purdue.edu/~aleix/aleix_face_DB.html
56. McArthur, L.Z., Baron, R.M.: Toward an ecological theory of social perception. Psychol. Rev. 3, 215–238 (1983)
57. Melville, P., Mooney, R.J.: Constructing diverse classifier ensembles using artificial training examples. In: International Joint Conferences on Artificial Intelligence, pp. 505–510 (2003)
58. Metz, C.E.: Basic principles of ROC analysis. Semin. Nucl. Med. 4, 283–298 (1978)
59. Mitsumoto, S.T., Kawai, H.: Male/female identification from 8 x 6 very low resolution face images by a neural network. Pattern Recognit. 2, 331–335 (1996)
60. Moghaddam, B., Yang, M.-H.: Gender classification with support vector machines. In: Sixth IEEE International Conference on Automatic Face and Gesture Recognition (FG), pp. 306–311 (2000)
61. Moghaddam, B., Yang, M.-H.: Learning gender with support faces. IEEE Trans. Pattern Anal. and Mach. Intell. 5, 306–311 (2002)
62. Mulford, M., Orbell, J., Shatto, C., Stockard, J.: Physical attractiveness, opportunity, and success in everyday exchange. American J. Sociol. 6, 1565–1592 (1998)
63. Nocedal, J., Wright, S.J.: Numerical optimization. Springer, New York (1999)
64. O'Toole, A.J., Abdi, H., Deffenbacher, K.A., Bartlett, J.C.: Classifying faces by race and sex using an autoassociative memory trained for recognition. In: 13th Annual Conference on Cognitive Science, Hillsdale, NJ, pp. 847–851 (1991)
65. O'Toole, A.J., Deffenbacher, K.A.: The perception of face gender: The role of stimulus structure in recognition and classification. Mem. and Cogn., 146–160 (1997)
66. Oja, E.: Subspace Methods of Pattern Recognition. Research Studies Press Ltd., Letchworth (1983)
67. Oja, E.: Principal components, minor components and linear neural networks. Neural Netw., 927–935 (1992)
68. Opitz, D., Maclin, R.: Popular ensemble methods: an empirical study. J. Artif. Intell. Res., 169–198 (1999)
69. Padgett, C., Cottrell, G.W.: A simple neural network models categorical perception of facial expressions. In: Proceedings of the 20th Annual Cognitive Science Conference, Madison, WI, pp. 806–807 (1998)
70. Phillips, P.J.: Support vector machines applied to face recognition. Adv. Neural Inf. Process Syst., 803–809 (1998)

71. Rosenberg, S.: New approaches to the analysis of personal constructs in person perception. In: Land, A.L., Cole, J.K. (eds.) Nebraska symposium on motivation, pp. 179–242. University of Nebraska Press, Lincoln (1977)
72. Rowley, H.A., Shumeet, B., Kanade, T.: Neural network-based face detection. IEEE Trans. Pattern Anal. and Mach. Intell. 1, 23–38 (1998)
73. Russell, S., Norvig, P.: Artificial intelligence: A modern approach. Prentice Hall, Upper Saddle River (2002)
74. Sirovich, L., Kirby, M.: Low dimensional procedure for the characterization of human faces. J. Opt. Soc. Am. 3, 519–524 (1987)
75. Swets, D.L., Weng, J.: Using discriminant eigenfeatures for image retrieval. IEEE Trans. Pattern Anal. and Mach. Intell. 8, 831–837 (1996)
76. The MathWorks, Using MATLAB: The language of technical computing. The Mathworks, Inc., Natick, MA (2000)
77. Todd, J.T., Mark, L.S.: The perception of human growth. Sci. Am. 2, 132–144 (1980)
78. Turk, M.A., Pentland, A.P.: Eigenfaces for recognition. J. Cogn. Neurosci. 1, 71–86 (1991)
79. Turk, M.A., Pentland, A.P.: Face recognition using eigenfaces. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Silver Spring, MD, pp. 586–591 (1991)
80. Valentin, D., Abdi, H., Edelman, B.E., O'Toole, A.J.: Principal component and neural network analyses of face images: What can be generalized in gender classification? J. Math. Psychol. 4, 398–413 (1997)
81. Valentin, D., Abdi, H., O'Toole, A.J.: Categorization and identification of human face images by neural networks: A review of the linear autoassociative and principal component approaches. J. Biol. Syst. 3, 413–429 (1994)
82. Valentin, D., Abdi, H., O'Toole, A.J., Cottrell, G.W.: Connectionist models of face processing: A survey. Pattern Recognit. 9, 1209–1230 (1994)
83. van der Heijden, F., Duin, R.P.W., de Ridder, D., Tax, D.M.J.: Classification, parameter estimation, and state estimation: An engineering approach using MATLAB. John Wiley & Sons, Ltd., Chichester (2004)
84. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
85. Wechsler, H., Gutta, S., Philips, P.J.: Gender and ethnic classification of Face Images. In: 3rd Int. Conf. on Automatic Face and Gesture Recognition, Nara, Japan, pp. 194–199 (1998)
86. Whitaker, C.J., Kuncheva, L.I.: Examining the relationship between majority vote accuracy and diversity in bagging and boosting (2003), `http://www.informatics.bangor.ac.uk/kuncheva/papers/lkcw_tr.pdf`
87. Zebrowitz, L.A.: Reading faces: Window to the soul? Westview Press, Boulder (1998)
88. Zebrowitz, L.A., Montepare, J.M.: Impressions of babyfaced individuals across the life span. Dev. Psychol. 6, 1143–1152 (1992)
89. Zebrowitz, L.A., Montepare, J.M.: Social psychological face perception: Why appearance matters. Soc. and Personality Psychol. Compass 3, 1497–1517 (2008)
90. Zebrowitz, L.A., Montepare, J.M., Lee, H.K.: They don't all look alike: Individuated impressions of other racial groups. J. Personal and Soc. Psychol. 1, 85–101 (1993)
91. Zenobi, G., Cunningham, P.: Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In: 12th Conference on Machine Learning, pp. 576–587 (2001)
92. Zhao, W., Chellappa, R., Phillips, P.J., Rosenfeld, A.: Face recognition: A literature survey. ACM Comp. Surv. 4, 399–458 (2000)

The Analysis of Crowd Dynamics: From Observations to Modelling

B. Zhan, P. Remagnino, D.N. Monekosso, and S. Velastin

Abstract. Crowd is a familiar phenomenon studied in a variety of research disciplines including sociology, civil engineering and physics. Over the last two decades computer vision has become increasingly interested in studying crowds and their dynamics: because the phenomenon is of great scientific interest, it offers new computational challenges and because of a rapid increase in video surveillance technology deployed in public and private spaces. In this chapter computer vision techniques, combined with statistical methods and neural network, are used to automatically observe measure and learn crowd dynamics. The problem is studied to offer methods to measure crowd dynamics and model the complex movements of a crowd. The refined matching of local descriptors is used to measure crowd motion and statistical analysis and a kind of neural network, self-organizing maps were employed to learn crowd dynamics models.

1 Introduction

We are interested in devising methods to automatically measure and model the crowd phenomenon. Crowded public places are increasingly monitored by security and safety operators. There are companies (for example LEGION) that have employed large resources to study the phenomenon and generate realistic simulations: for instance, to optimize the flow of people in a public space. Section 2 presents some details about crowd related work, including the applications, research in computer vision and research in other areas, like civil engineering and sociology. The purpose of Section 2 is to give an overview of the state-of-the-art on crowd analysis and to discuss the probability to bridge the research from computer science to areas like civil engineering and sociology.

B. Zhan, P. Remagnino, D.N. Monekosso, and S. Velastin
Kingston University, Penrhyn Road, Kingston-upon-Thames, Surrey, KT1 2EE, UK
e-mail: B.Zhan@kingston.ac.uk

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 441–472.
springerlink.com

Computer Vision research offers a large number of techniques to extract and combine information from a video sequence acquired to observe a complex scene. The life cycle of a computer vision system includes the acquisition of the monitored scene with one or more homogeneous or heterogeneous cameras, the extraction of features of interest and then the classification of objects, people and their dynamics. The overall objective of our work is to develop an intelligent crowd analysis system, in this chapter the work includes combining techniques from image processing, machine vision, statistics and neural networks to measure and model crowd dynamics. In particular, novel methods to measure crowd dynamics are proposed using image processing and machine vision techniques and two crowd modelling methods are developed by statistical methods and neural networks.

In simple scenes the background is extracted with statistical methods and then foreground data and related information are inferred to describe and model the scene. Background is usually defined as stationary data, for instance man made structure, such as buildings, in a typical video surveillance application, or the indoor structure of a building in a safety application, for instance deployed to monitor and safeguard elderly people in a home. Unfortunately, background modelling becomes rapidly less effective in complex scenes and its usefulness seems to be inversely proportional to the clutter measured in the scene. Figure 1 shows a small experiment testing the effectiveness of background modelling with different types of scenes. Three frames per chosen sequence and the resulting background image built with roughly 1000 frames are illustrated. The background modelling works well with the first scene; it fails to recover the background of some regions in the second scene because of the frequent occupancy over these regions; and in the third scene, due to the continuous clutter, the background model can be barely recovered. When the monitored scene becomes very cluttered, then one could think of

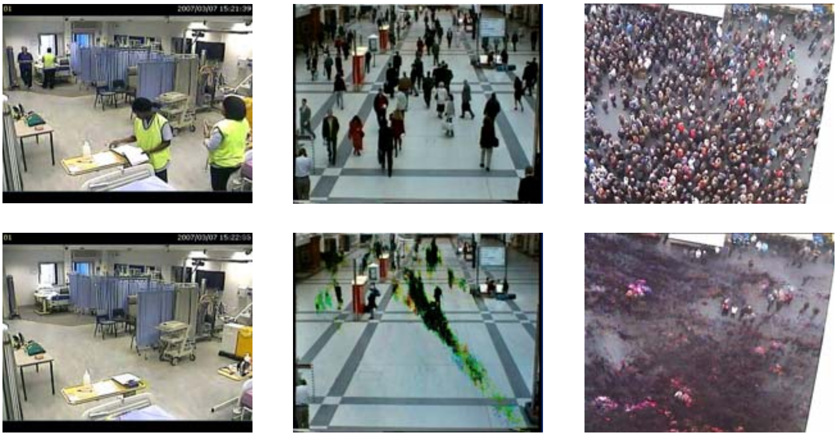


Fig. 1 The example frames and the built background images from three different scenes. Left to right: three different scenes; top to bottom, three example frames and the built background images, respectively

measuring dynamics with optical flow methods, designed to extract information about the dynamics of the scene, typically using gradient information. Unfortunately, popular and conventional optical flow techniques such as Horn and Schunck [36] and Lucas and Kanade [60] also work poorly with heavily crowded scenes. On the other hand, feature based optical flow techniques using multi-resolution work quite well with relatively high frame rate (typically around 25fps) video sequences [15]. Section 3 presents two methods that can automatically measure crowd dynamics. The methods are feature based and employ more sophisticated constraints. They are briefly presented in the chapter and for more details the reader is referred to [98] [100]. Both methods have been assessed with video sequences capturing different types of crowded situations. A comparison of the two methods was carried out and also described in the chapter, for more details the reader should refer to [99]. The performances of both methods produce satisfactory results, even with low frame rate video sequences (typically 4 to 8 fps).

> Optical flow or optic flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer (an eye or a camera) and the scene. In the survey of Beauchemin [11] existing optical flow techniques are investigated, including: 1) differential methods; 2) frequency based methods; 3)correlation based method; 4)multiple motion methods and 5) template refined methods.

Section 4 describes the methods used to model crowd dynamics. First a statistical method is introduced. This method is focused on defining the main path of the crowded scene [95]. Then a neural network based approach is proposed to capture the crowd dynamics with a reduction of the dimensions of the input data. The self-organizing map technique is employed for this purpose and the results have been generated for different types of crowded scenes. Section 5 discusses the obtained results and sheds some light on the future directions of the work on crowd analysis.

2 Background

The steady population growth, along with the worldwide urbanization, has made the crowd phenomenon more frequent. It is not surprising; therefore, that crowd analysis has received attention from technical and social research disciplines. The crowd phenomenon is of great interest in a large number of applications:

Crowd Management: Crowd analysis can be used for developing crowd management strategies, especially for increasingly more frequent and popular events such as sport matches, large concerts, public demonstrations and so on, to avoid crowd related disasters and insure public safety.

Public Space Design: Crowd analysis can provide guidelines for the design of public spaces, e.g. to make the layout of shopping malls more convenient to costumers or to optimize the space usage of an office.

Virtual Environments: Mathematical models of crowds can be employed in virtual environments to enhance the simulation of crowd phenomena, to enrich the human life experience.

Visual Surveillance: Crowd analysis can be used for automatic detection of anomalies and alarms. Furthermore, the ability to track individuals in a crowd could help the police to catch suspects.

Intelligent Environments: In some intelligent environments which involve large groups of people, crowd analysis is a pre-requisite for assisting the crowd or an individual in the crowd. For example, in a museum deciding how to divert the crowd based on to the patterns of crowd.

Crowd management and public space design are studied by sociologists, psychologists and civil engineers; virtual environments are studied by computer graphics researchers; visual surveillance and intelligent environments are of interest to computer vision researchers. The approach favoured by psychology, sociology, civil engineer and computer graphics research is an approach based on human observation and analysis. Sociologists, for instance, study the characters of a crowd as a social phenomenon, exploring human factors. For example, the computational model developed by Seed Projects at Stanford University [86], incorporated human behaviour in environments with emergency exits. The Crowd - MAGS Project, which is funded by GEOIDE and the Canadian Network of Centers of Excellence in Geomatics, aims to develop micro-simulations of crowd behaviours and the impact of

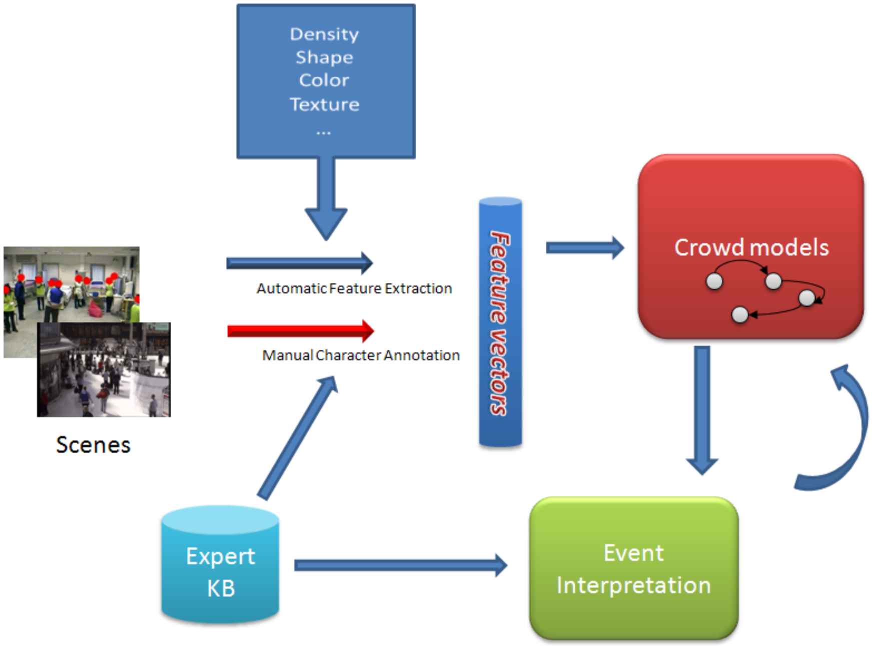


Fig. 2 A framework for Crowd analysis

Table 1 Features in crowd analysis by computer vision methods

Sensor typology and topology	Moving or Static platform Number of cameras Type of video sequence: colour or gray scale, etc.	
Environmental conditions	Indoor/outdoor Level of clutter Light condition, etc.	
Scene typology	Individual characters Appearance, etc.	location/velocity/etc.
	Collective Average speed, etc.	Crowd density

police or military groups [22]. The Police Academy of the Netherlands and School of Psychology of University of Liverpool are cooperating on a project funded by the UK Home Office: “A European study of the interaction between police and crowds of foreign nationals considered to pose a risk to public order” [1].

On the other hand, computational methods, such as those employed in computer graphics and vision methods, focus on extracting quantitative features and detecting events in crowds, synthesizing the phenomenon with mathematical and statistical models. For example, an early project funded by the EPSRC in the UK was concerned with measuring crowd motion and density and hence potentially dangerous situations [25] [87] [93]. The EU funded projects PRISMATICA [75] and ADVISOR [2], completed in 2003, were concerned with the management of public transport networks through CCTV cameras. The UK EPSRC funded project BEHAVE, was concerned with the pre-screening of video sequences with the detection of abnormal or crime-oriented behaviour [12]. ISCAPS [42] started in 2005, a consortium of 10 European ICT companies and academic organizations, aims to provide automated surveillance of crowded areas. SERKET, a recently started EU project aims to develop methods to prevent terrorism [40].

Figure 2 illustrates the processes involved in crowd analysis. In a crowd scene the attributes of importance are crowd density, location, speed, etc. This information can be extracted either manually or automatically using computer vision techniques. Crowd models can then be built based on the extracted information. Event discovery is achieved using pre-compiled knowledge of the scene or using the computational model, although both approaches can be combined. In both cases the model is updated with newly extracted information.

2.1 *Crowd Information Extraction*

The components of crowd analysis from a computer vision perspective are described in Table 1. Essentially, the sensors and their topology influence the scene capture processes; environmental conditions, such as natural and artificial illumination changes often introduce noise; the scene typology affects the type of process one requires to extract the most accurate information of a dynamics scene.

Visual surveillance methods have been developed to estimate the motion of objects and people in the scene, in isolation or in groups; a review can be found in [37]. When video is analysed for very crowded scenes, conventional computer vision methods are not appropriate, in these cases methods must be designed to cope with extreme clutter. Features from conventional image processing are still employed, such as colour, shape and texture etc. However, sophisticated methods have been developed to retrieve crowd information. In the following sections the existing state-of-the-art will be reviewed.

2.1.1 Density Measurement

An important crowd feature is crowd density and it is natural to think that crowds of different densities should receive a different levels of attention.

Research methods have been proposed for crowd analysis which employ background removal techniques such as [93], [61] and [26]. These studies makes use of examples to map the global shape feature directly to configurations of humans, and work under the typical assumption that the number of foreground pixels are proportional to the number of people, which is only true when there are not serious occlusions between people.

Image processing and pattern recognition techniques are also used for the analysis of the scene to estimate the crowd density. Marana et al. [64] assume that images of low-density crowds tend to present coarse texture, while images of dense crowds tend to present fine textures. Self-organizing neural maps [65] combined with Minkowski fractal dimensions [63] are employed to deduce the crowd density from the texture of the image. The work by Marana is compared in [76] with another method that uses Chebyshev moments. An optimization of performance under different illumination conditions is discussed. Lin et al. [59] present a system that estimates the crowd size through the recognition of the head contour using Haar wavelet transform (HWT) and support vector machines (SVM).

Alternative methods combine several techniques, to achieve more accurate and reliable measurements. For example, in [87], an edge-based technique is integrated with background removal using a Kalman filter. Marana et al. [62] use different methods including Fourier and Fractal analysis and classifiers to estimate the crowd density level. Kong et al. in [52] [53] employ background subtraction and edge detection; the work defined the extracted edge orientation and blob size histograms as features. The relationship between the feature histograms and the number of pedestrian is learned from labelled training data. Obviously more cues may indicate a more accurate solution.

2.1.2 Recognition

Conventional visual surveillance focuses on object detection and tracking. In essence, image processing techniques are employed to extract the chromatic and shape information of the moving objects and the background for detecting and tracking purposes.

For crowd dynamics modelling, detecting and tracking are also important as they provide the location and velocity features of the dynamics. Crowded scenes add a degree of complexity to the conventional detection and tracking problem of single individuals. In the following sections the focus will be on methodologies for crowded situations.

Face is the most discriminating feature of the human body, and many researchers try to detect a pedestrian through face detection. The majority of the existing research employs supervised learning methods to detect faces in a crowded situation, for example [85] [58] [43][38].

Pedestrian detection and tracking is a well studied problem in computer vision. Many methods have been proposed, such as using the afore mentioned background removal technique, or combining chromatic and shape information of the tracked pedestrians. The following sections discuss the methods that try to provide a solution for pedestrian detection in crowded scenes.

Occlusion caused by the high clutter of the pedestrian in a crowd scene is the major challenge for crowd detection problems. Research is being carried out to address the problem by using human body parts, for example [91] [28] [57]. Besides conventional cues of pedestrian appearance, space-temporal cues are also used for detection. Brostow et al. [17] tackle the problem by tracking simple image features and probabilistically grouping them into clusters representing independently moving entities. In extremely cluttered scenes, individual pedestrian cannot be properly segmented in the image. However, sometimes the *crowd* within which the pedestrians share a similar purpose can be recognized. Reisman et al. [79] propose a scheme that uses slices in the spatial-temporal domain to detect inward motion as well as intersections between multiple moving objects. The system calculates a probability distribution function for left and right inward motion and uses these probability distribution functions to infer a decision for crowd detection.

2.1.3 Tracking

Tracking has been proposed to localize the interested object in time-space. Also the velocity feature can be derived afterwards. Though as a natural extension of detection, tracking has its own problem to recognize and identify pedestrians in the consecutive frames. Tracking could be regarded as the most popular topic in visual surveillance, however currently for crowd analysis, most of the techniques are validated only for multiple (e.g. up to 10) people.

As discussed in the last subsection, occlusions can occur very frequently when there are many objects and people in the scene. Tracking techniques have to overcome this problem in order to continuously track before, during and after the occurrence of occlusions. A comprehensive review on occlusion handling can be found in [30]. A formulation of the occlusion problem is provided, and the techniques are divided in two groups: the merge-split approach, which addresses the problem to re-establish object identities following a split, and straight-through approaches, which maintain object identities at all times.

Crowd scenes increase the complexity of tracking because there are multiple moving objects in the scene. Different techniques are developed to improve the continuous tracking of an individual in a crowd.

- **Likelihood.** Colour, edge etc. are the most popular features in tracking. In crowds, salient traceable image features are of particular interest. For example, as one of the good candidates, interest points (IPs) are employed in [30] and [67].
- **Human body model.** Methods using models of human bodies or human body parts have been developed for tracking in complex crowded scenes, which are usually completed with probabilistic frameworks, examples like Zhao [101][102] [92] [91].
- **Tracking inference strategies.** Tracking inference strategies have been developed for the problem of tracking multiple objects. For non-linear and non-Gaussian dynamic models, a particle filter technique, also known as CONDENSATION [41], is one of the most popular among those. Particle filters are sequential Monte Carlo methods based upon a point mass (or 'particle') representations of probability densities [27]. Large portions of multiple object tracking work have employed this technique, for example [88][73] [80] [18][51] [44].
- **Data assoication.** To address data association problems, there are Multiple Hyphotheses Tracker (MHT) and Joint Probabilistic Data Association Filter (JPDAF). MHT tries to keep the track of all the possible hypotheses over time [78]. A detailed summary and a discussion of MHT for multiple target tracking is included in [13]. JPDAF computes a Bayesian estimation of correspondence between the different features and the different objects, e.g. [77] [45].

In certain cases, interaction happens frequently in crowded scene. Researchers have shown great interest in studying these interactions to get new perspectives on tracking techniques. For example, both Smith et al. [81] and Khan et al. [47] propose to use Markov Chain Monte Carlo (MCMC) and the particle filter. Some researchers interpret interactions as relationships between pedestrians and a group (pedestrian merging/splitting into groups) [66] [69].

Furthermore, for large public areas the use of a multi-camera system is required to cover most of the monitored areas, for example [70] [20][46] [48].

2.2 *Crowd Modelling and Events Inference*

Dynamics in public spaces can indeed be recurrent. Crowd information can be better exploited to indicate the status of the crowd so that crowd events can be inferred. Crowd models have been built to represent these statuses, either implicitly or explicitly. On the other hand, some research makes direct use of crowd information instead of building models. In such cases, the events are usually inferred based on some prior knowledge of the properties of the particular scene and the crowd. In this section, crowd models and events inference in computer vision will be presented as well as some crowd models from non vision areas.

2.2.1 Crowd Models and Crowd Events Inference in Computer Vision

In computer vision crowd modelling is achieved based on the extracted information from visual data and normally can be employed in crowd events inference. Meanwhile, there are also some approaches that attempt to infer events without the construction of models.

- **Crowd models as representations of recurrent behaviours.** Zhan et al. [94] Andrade et al. [6][5][7] characterize crowd behaviour by observing the crowd optical flow associated with the crowd, and use unsupervised feature extraction to encode normal crowd behaviour.
- **Event inference.** Early work on crowd monitoring and crowd event inference using image processing is reviewed by Davies et al. [25]. More recent work on this includes [14] [68] [24] [23] [19]. In these methods especially, assumptions of crowd are usually involved, indicating that some prior knowledge is required for events inference.

2.2.2 Crowd Models from non Vision Approach

Computational models aim at describing and predicting the collective effects of crowd behaviour by identifying the relationship between crowd features.

- **Physics inspired models.** Several quantitative factors of crowds and pedestrians are measurable. This fact encourages researchers to look for the mathematical models of crowd dynamics. For example Helbing [34][35][33] propose social force model based on the social field theory. Hughes [39] describes the crowd by "types", where pedestrians in each type have the same walking habits.
- **Agent based models.** These are qualitative models that employ fuzzy methods to describe the relations of factors and crowd motion, instead of using pure mathematical methods. Agent-based models use agents to represent the pedestrian or the crowd, examples include [72][74] [16]. Some work on agent-based models has already been commercialised, such as the work of Keith Still at Crowd Dynamics Ltd [21] and LEGION international LTD [56], both provide pedestrian simulations for space design and planning, based on agent technology.
- **Cellular automation models.** Another research approach employs the construction of local models, where the active area has been virtually divided into cells, such as [3] [54].
- **Nature based models.** Some of the models take their inspiration from nature. The emotional ant model [10] extends the psychological information using a biologically inspired ant agent as a crowd and Kirchner et al. [49] apply a bionics approach to the cellular automation model.

2.3 Examples of Bridging the Research

Computer simulation can be used to evaluate the developed system's performance. Considering that real visual evidence for abnormal scenarios are rare or unsafe to reproduce in a controllable way, Andrade et al. [4] have developed an approach generating simulations to allow training and validation of computer vision systems applied to crowd monitoring. The simulation is generated by a pedestrian path model and a pedestrian body model. Vu et al. [90] conceive a test framework that generates 3D animations corresponding to behaviours recognised by an interpretation system. In other words, this is a test system for a given interpretation system for generating test animations. Traditional models can be borrowed for computer vision analysis. Anotonini et al. [8][9] propose a framework using discrete choice model, which is widely used in traffic simulations, for pedestrian dynamics modelling.

The work of traditional analysis shows that all of the factors or information extracted from the real world using computer vision techniques is inter-related. Moreover, researchers have proposed the probable relationships in their work, which represent the human understanding of crowd dynamics. On the other hand, computer vision techniques have the ability of exploiting the special environmental constraints, which could be applied to calibrate the proposed models. We can claim that it is possible that to develop intelligent systems combining these works with computer vision approaches. The system would be capable of automatically understanding and modelling the crowd behaviours at both instantaneous and recurrent level.

3 Measuring Crowd Motion

Algorithms exist to analyze simple scenes, where a few people enter and exit the field of view of the deployed cameras. In such scenes, people and objects are identified and tracked throughout the network of cameras. People and objects, such as vehicles, are tracked between frames[1] and their trajectories are also predicted using conventional Kalman filters, or more sophisticated particle filter techniques. We studied algorithms that use refined matching methods, exploiting local descriptors to derive the dynamics features instead of providing a conventional *tracking* of pedestrians. The problem with tracking in very cluttered and complex scenes is that matching is not always possible and tracks are frequently lost, creating fragmentation in the tracking process. What we propose is tracking for short periods of time, and we provide two algorithms to provide robust matching between frames for use in short-time tracking. The extracted and matched dynamics features can then be directly used in the process of crowd understanding and dynamics modelling.

3.1 Method 1*: Pyramid-Based Interest Points Topological Matching*

In order to devise algorithms to automatically derive complex crowd dynamics, local descriptors, classified as interest points have been extracted using colour gradient information at scale space. Furthermore, besides the use of the extracted descriptors,

[1] Tracking refers to matching and predicting position and form of extracted features between time frames.

an advanced matching improved by incorporating topological constraints has been developed.

3.1.1 Extraction of Local Descriptor: Harris Detector

The first method employs a modified version of the Harris interest point detector [31]). The Harris interest point detector provides a repeatable and distinctive descriptor of the image features and it is view-point and illumination invariant. This detector extracts feature points, making use of the three chromatic channels is defined as the M matrix:

$$M = G(\sigma) \otimes \begin{pmatrix} C_x \cdot C_x & C_x \cdot C_y \\ C_y \cdot C_x & C_y \cdot C_y \end{pmatrix} \tag{1}$$

In the operation the image is firstly smoothed using a standard Gaussian operator (of deviation σ). C_x and C_y are respectively the gradient in x and y directions of the pixel chromatic triplet. They are estimated by applying the Gaussian derivative operator $G(\sigma)$ of (deviation σ) to the smoothed image, this is efficiently implemented by using the method from [89]. The interest points are then extracted using term R, which is calculated as a combination of the Eigen values of the M matrix:

$$R = det(M) + \kappa trace^2(M) \tag{2}$$

Where κ is a constant where $00.4 \leq \kappa \leq 0.06$. The points with local maximum are selected as interest points. A multi-scale approach is used, generating the interest points at the lowest (finest scale) layer and then projecting them up to the top (coarsest scale) layer of the generated pyramid.

3.1.2 Point Matching

The matching is carried out in two steps: searching for the candidate matching points by similarity, and then applying the topological constrains described later. Frequent

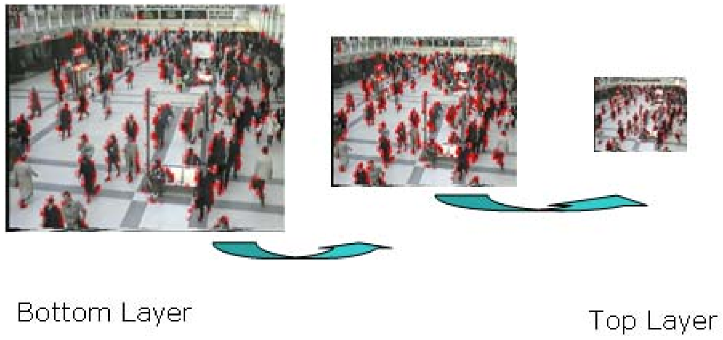


Fig. 3 Interest Point Generation, from bottom layer to top layer

occlusions reduce the probability of identifying correct matches, as a result without local support, similar gradient local regions might be found as plausible matches, generating false positives. In this implementation a topological constraint is proposed to make the search for correspondences more robust. Gabriel [29] suggested a similar method using topological information, however in his algorithm the area (object) of interest was predefined and the topological information was evaluated by the already know centre of the object. In this approach, instead of detail tracking a particular object over long period, the motion of two consecutive frames is more desirable. Therefore, the necessary local support is derived from local windows centered at the interest point and the relative location of the interest points in such windows is used. Support is estimated for the matched interest point pair inside the support window.

3.1.3 Temporal Pyramidal Analysis

Temporal smoothing and matching is also carried out by comparing a number of N spatial pyramids, corresponding to a specific time window. Thus, a spatial-temporal pyramidal analysis of the sequence is generated for a number of frames. Temporal smoothing is employed to enforce time consistency on matches, reducing false alarms generated by unstable interest points.

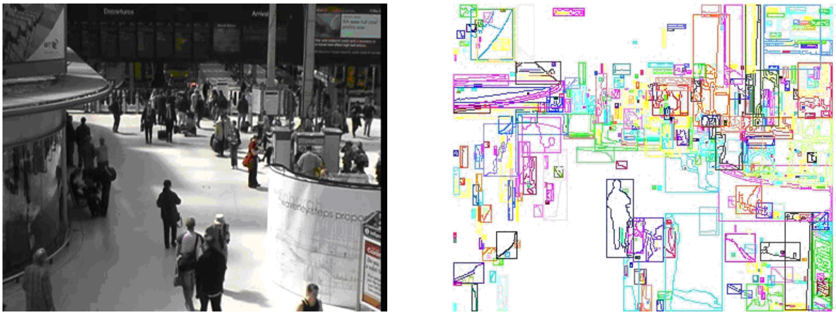
So matching is carried out in both space and time, starting at the highest level (coarsest level) of each pyramid, searching interest point correspondences between the initial frame of the N frames and all the other frames within the given time period (corresponding to $N-1$ matches). Spatial matching works from the top (finest scale) of a pyramid to the bottom (coarsest level). Then temporal integration of pyramidal matches of interest point j in 0^{th} frame can then be applied by combining the N matches.

3.2 Method 2: Using Edge Continuity Constrains of Interest Points

The second method is developed using local descriptors, but also incorporating shape information. Inspired by the methodology used in deformable object tracking, edge information is extracted and descriptor points are extracted as points along an edge with local maximum curvature. The information about an edge is maintained and used to impose the *edgelet constraint* and refine the estimate. Thus, the advantage of using point features which are flexible to track, and the advantage of using edge features which maintain structural information, are combined here.

3.2.1 Edge Retrieval

The Canny edge detector is employed to extract the edge information of a given frame. Each Canny edge is a chain of points, and all the edges are stored in an edge



(a) Original frame.

(b) Edge chains and their bounding boxes).

Fig. 4 Edge Chain

list. Figures. 4 show an example image frame and the extracted edge chains with associated bounding boxes, respectively. It can be observed that even in a scene which depicts a crowd of moderate density, edge chains can occlude each other, increasing the descriptor matching complexity.

The Canny edge detector is an approach which is optimal for a step edge corrupted by white noise. The optimality of the detector is related to three criteria. The detection criterion is about low error rate. It is important that edges occurring in images should not be missed and that there be no responses to non-edges. The second criterion is that the edge points be well localized. The distance between the edge pixels as found by the detector and the actual edge is to be at a minimum. A third criterion is to minimize multiple responses to a single edge. Thus, based on these criteria, the Canny edge detector was proposed and has become one of the most popular edge detectors [82].

3.2.2 Curvature Estimation and Interest Point Extraction

Interest points can be quickly extracted for a sequence of frames, for instance with the Harris corner operator used in the last section. However Harris interest points can only represent the local characteristics of an image in isolation, while the shape information of the moving person/people is lost. In this implementation the interest points are from the edges and then the constraint is imposed that they lie on a specific edge. Each edge can be represented by a parameterized curve:

$$x = x(t), \tag{3}$$

$$y = y(t). \tag{4}$$

The curve is smoothed with a Gaussian filter, as follows

$$X(t) = G(t) \otimes x(t), \tag{5}$$
$$X'(t) = G'(t) \otimes x(t), \tag{6}$$
$$X''(t) = G''(t) \otimes x(t). \tag{7}$$

The curvature of each edgelet can then be given by [71] :

$$\kappa = \frac{X'Y'' - Y'X''}{(X'^2 + Y'^2)^{\frac{3}{2}}} \tag{8}$$

Corner points are defined and extracted as the local maxima of the absolute value of curvature on each edge. Thus the edge representation is changed from a point sequence to a corner point sequence, resulting in a list of corner point sequences for all the edges of the image.

3.2.3 Point Matching and the Edgelet Constraint

Given two consecutive frames I_t and I_{t+1}, the motion is estimated for each extracted point of interest. For each corner point with coordinate (x,y) in I_t a rectangular search window is defined centering at (x,y)in I_{t+1}. A look-up table (LUT) contains corner points and edge information is generated to enhance the matching. The correspondence is matched by using curvature information of corner points in the search window in LUT against the reference point. The error is calculated by the curvature.

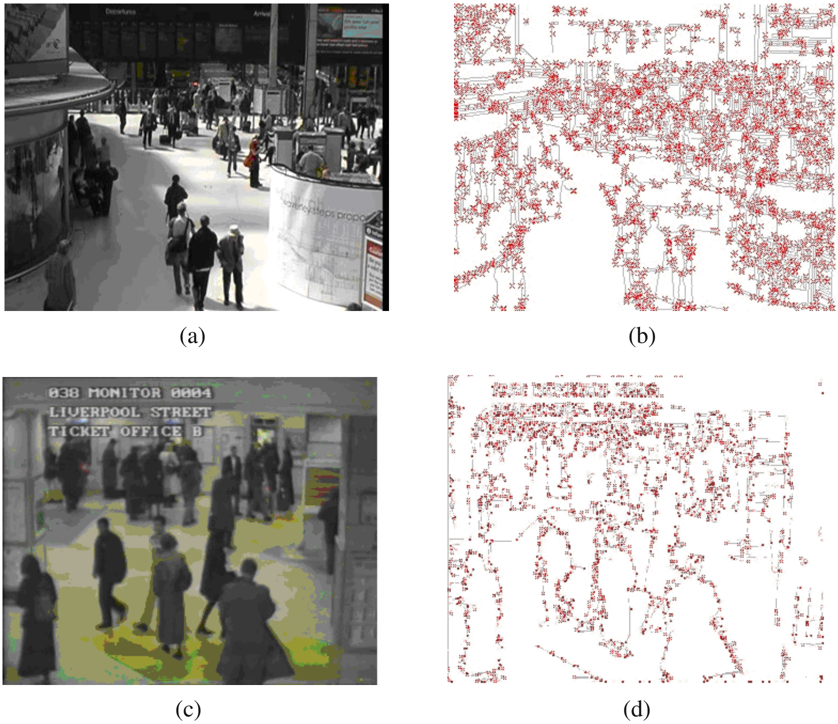
Complex dynamics and frequent occlusions generated in crowd scenes make the estimation of motion a very complex task. Point matching in isolation is too fragile and prone to errors to provide a good motion estimator. If the interest points are extracted on edge chains, then the edge constraint can be imposed and used.

For an image frame I_t, every edge is split to a uniform length edgelets represented by sub-sequences (so called edgelet). There are two reasons for doing this: to avoid a very long edge that could be generated by several different objects, and to enhance the matching of the edge fragments that are generated by occlusions. For each corner point there are n candidate matching points. Each candidate point belongs to an edgelet, thus there are $m(m <= n)$ candidate matching edgelets. To find the best match, three parameters are used: energy cost, variation of displacements and the match length for each candidate, and these are combined into a single matching score. The length of the edgelet is assumed to be sufficiently small so that it would not split again to two or more matches. This is so that their candidate points correspond to the same candidate sequence.

The matching is carried out over every point of the given edgelet, and an overall matching will be examined to determine the matched edgelet.

3.3 Comparison of the Two Methods

When the scene is very complex, occlusions make it virtually impossible, not only to track individuals, but also to estimate a stochastic background model. The two



(a) (b)

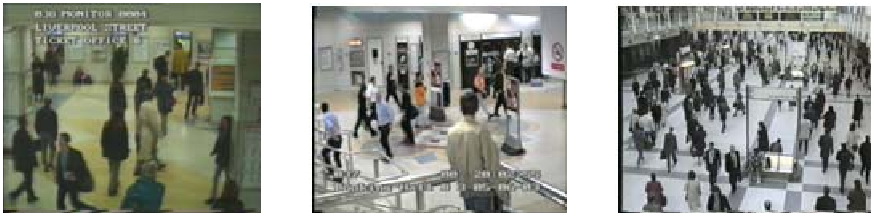
(c) (d)

Fig. 5 Two scenes of different complexity levels are illustrated. The original frames (left) and the extracted corner points (right) which are marked with red crosses on grey edges

described motion estimation methods were validated and compared. In both of the algorithms, constraints are applied to improve the robustness of the matching between individual descriptors. The first algorithm carries out a local check of the spatial temporal consistency of the colour gradient, supported by the local topology constraints, and the second one uses the points of local extreme curvature along Canny edges and applies contour constraints.

3.3.1 Testing Data

The two motion estimation algorithms are tested using three sequences taken from crowded public space, and quantitative results are generated. In the following a brief description of the test dataset used in the experiments is given. Then the details of the testing methods adopted, and an explanation of the results generated from the tests are introduced. Again additional visual results are included at the end of the section. Sample frames from the three sequences are shown in Figure 6: sequence 1 (left) is a mid field scene with people scattered across the field of view; sequence 2 (middle) is a mid field scene with major motions taking place in certain areas; sequence 3 (right) is a far field scene with pedestrians present in all parts of the field of view, with some predominant trajectories.



(a) (b) (c)

Fig. 6 Sample frames from 3 testing sequences

3.3.2 Testing Based on Local Descriptors

In this test only the quality of matching of individual local descriptors is considered. For each pair of consecutive frames, local descriptors in the initial frame are compared with their corresponding local descriptors, found by the two presented algorithms, in the target/second frame, respectively. Two measures, Mean Similarity (MS) and Mean Absolute Error (MAE), are used here.

The images in Figure 7 represent the plots of MS and MAE for the two algorithms tested against the three sequences. MS and MAE are calculated every frame along the sequence. In each plot the *x* axis represents time (the number of the frame) and the *y* axis represents the values of MS and MAE, respectively. Hence, for the two algorithms the MS and MAE for the three testing sequences are both good, though in most of the cases the second algorithm has a higher MS and a lower MAE. Also, along the time scale the MS and the MAE produced from the first algorithm fluctuate a lot while the second one produces more stable results. It can be concluded that the second algorithm has a more desirable performances than the first one.

3.4 Testing Based on Motion Connect Component

The test here makes use of a connected components algorithm based on motion vectors (so called MCC – Motion Connected Component). The algorithm groups together motion vectors that are in close proximity and have common motion properties. The result of the MCC algorithm segments the motion field into clusters of uniform motion group (e.g. a (part of) pedestrian or a group of pedestrians), and the test is based on each MCC to assess the two algorithms. In order to assess the two algorithms with MCC, two measures, which are from evaluating search strategies: Recall and Precision, are adapted here.

For every frame an average Recall value and an average Precision value are calculated. Figure 8 gives the plots of Recall and plots of Precision; the layouts of these plots remain similar to the previous ones, though the *y* axis represents Recall and Precision, respectively.

From the plots again the results of Recall and Precision of both of the algorithms, especially the results of Recall, are satisfied. It can be observed that the results of

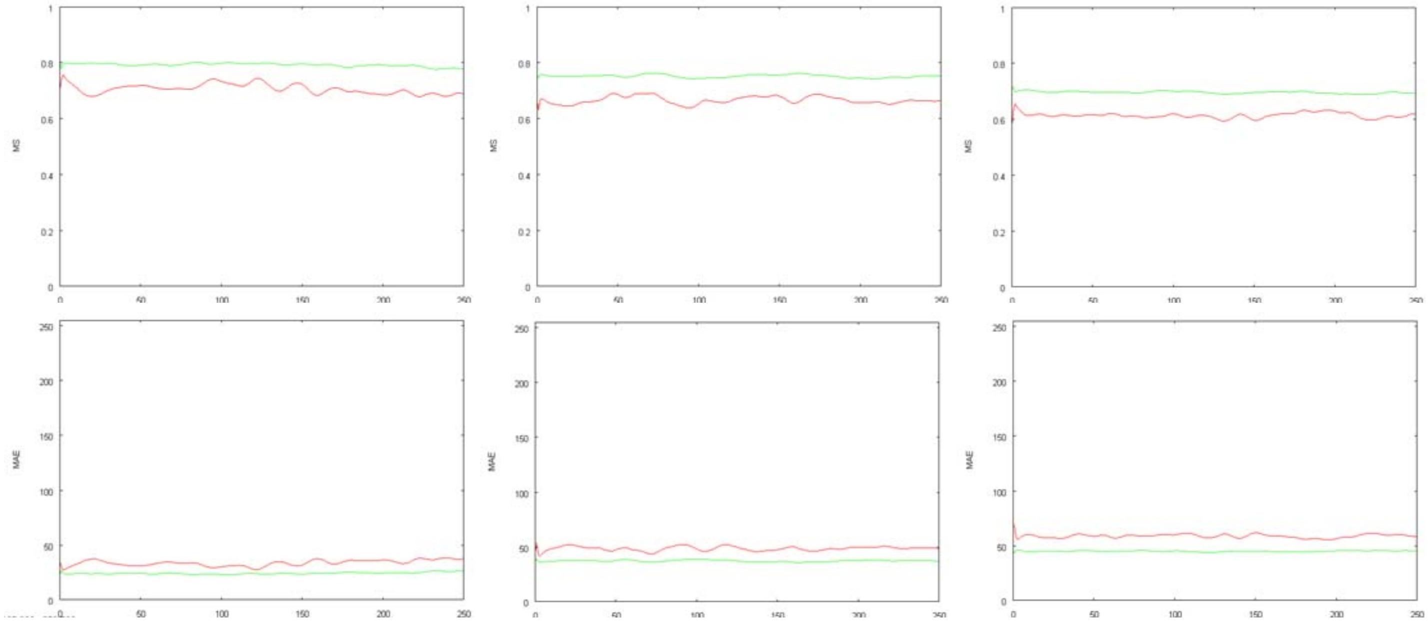


Fig. 7 MS (top row) and MAE (bottom row) along time for the 3 testing sequence (From top to the bottom: sequence 1, sequence 2 and sequence 3), red lines for Algorithm 1; green lines for Algorithm 2. Algorithm 2 keeps higher in MS and lower in MAE

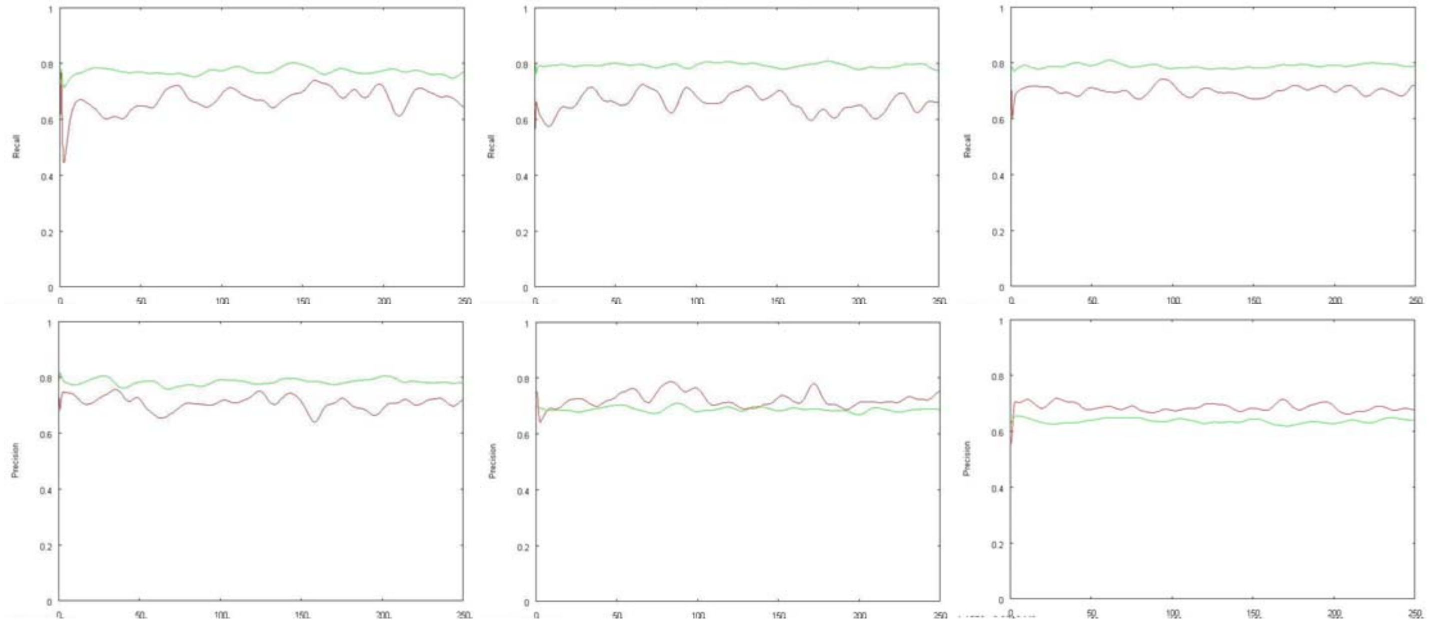
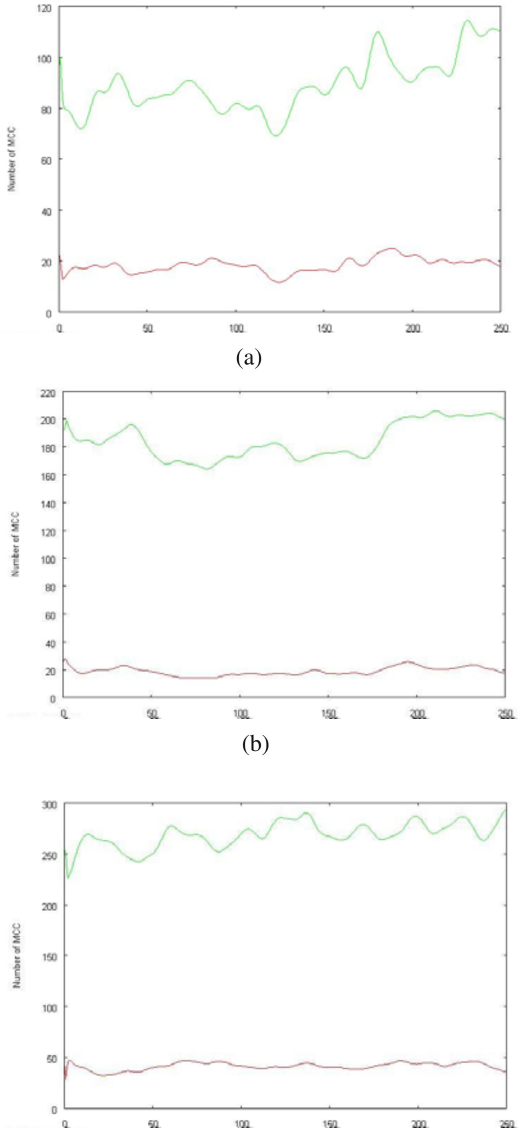


Fig. 8 Recall (top row) and Precision (bottom row) along time for the 3 testing sequence (From top to the bottom: sequence 1, sequence 2 and sequence 3), red lines for Algorithm 1; green lines for Algorithm 2. Algorithm 2 has higher values of Recall



(a)

(b)

(c)

Fig. 9 Number of MCCs along time for the 3 testing sequence, red lines for Algorithm 1; green lines for Algorithm 2 (From top to bottom: sequence 1, sequence 2 and sequence 3). Algorithm 3 detects much more MCCs for all of the three video sequences

Precision for sequence 3 is lower than the other three. One possible reason could be that, as sequence 3 is a far field view for a crowded scene, when mapping the bounding box of the MCC to the second frame, local descriptors of other MCC could be included and noise could be introduced.

When comparing the results of Recall, it can be seen that values for Algorithm 2 are always higher, though for sequence 2 and sequence 3 Precision values for Algorithm 1 are slightly higher. Here another measure should be taken into consideration, which is the number of the MCC detected by each algorithm. According to the plots in Figure 9, in sequence 1 the average number of MCC detected by Algorithm 1 is around 20, while by Algorithm 2 the number is around 100; in sequence 2, the numbers are around 20 and 200, respectively; in sequence 3 the numbers are around 40 and 280, respectively. Algorithm 2 detects much more MCC, especially for sequence 2 and 3. Due to the above fact and the fact Algorithm 2 produces higher Recall, it can be deduced that the slight drawback of the Precision only indicates more noise has been introduced to the assessment.

4 Modelling Crowd Dynamics

Crowds appear to move at random in a scene. In fact, this is not exactly true: people move purposively and their movements are guided by intentions. For instance, in a railway station or at an airport, people tend to enter and exit the scene at the gates and usually stop in front of a timetable, a shop or a cash point. Although at first chaotic, the video of a crowded place, if observed attentively, reveals main trajectories. We have studied two methods to extract the main paths or directions of motion of a crowded scene. They are described in the following sections of the chapter.

4.1 Statistical Analysis

The proposed method can be summarized in the following steps:

- Occurrence PDF(Probability Density Functions): foreground detection, connected components, accumulator,
- Orientation PDF(Probability Density Functions): correlation matrix, accumulator of block matching,
- Path discovery: previous orientation, probability calculation, path split.

4.1.1 Occurrence PDF

It is unrealistic to precompile a background model of a complex real world scene, such as those video recorded by security cameras in public spaces. This is because of sudden or continuous changes in illumination, shadows and noise in the video signals. This method assumes that the scene is not too crowded and the Gaussian mixture model [83] is used to build a robust model of the background of the scene. The foreground data is further processed to reduce noise. In particular, connected components have been implemented. Connectivity of foreground pixels gives more robustness to the foreground data and assures that only large foreground blobs are accepted for further analysis, while smaller blobs are rejected as likely noise.

A background model is essential for video analysis, to separate foreground data from the scene. There is a standard background adaptation carried out by averaging images over time, creating a background approximation which is similar to the current static scene except where motion occurs. While this is effective in situations where objects move continuously and the background is visible a significant portion of the time, it is not robust for scenes with many moving objects, particularly if they move slowly. A Gaussian mixture model is proposed in [83]. It is an adaptive tracking system that is flexible enough to handle variations in lighting, moving scene clutter, multiple moving objects and other kinds of changes to the observed scene. Rather than explicitly modelling the values of all the pixels as one particular type of distribution, the values of a particular pixel are simply modelled as a mixture of Gaussians. Based on the persistence and the variance of each of the Gaussians of the mixture, the model determines which Gaussians may correspond to background colours. Pixel values that do not fit the background distributions are considered foreground until there is a Gaussian that includes them with sufficient, consistent evidence supporting it.

For each frame foreground, features are accumulated for every pixel, so that after a relatively long video sequence the accumulator of the foreground occurrence throughout the whole image will have some information.

4.1.2 Orientation PDF

The image plane is segmented into a regular grid of cells ($N \times M$). The dimension of each cell is a multiple of 2 and each cell is square-shaped ($K \times K$). The idea is to speed up the matching process employed as a coarse estimator of motion between frames. Motion is estimated between consecutive frames, using the foreground blocks of the first frame as a reference and searching for an optimal match in the second frame. In the current implementation, block matching is carried out in a $3x3$ neighborhood, around the selected foreground cell. A cell is labelled as foreground if the majority of its pixels are indeed foreground. Matching performance is improved by matching only between foreground cells, ignoring background cells.

Each cell is therefore associated with a histogram representing the eight possible directions of motion. The intention here is to build a local representation of motion, similar to a discrete reinforcement learning technique [84], where each cell of the table has associated a quality array, indicating the likelihood of transition from the current cell to a neighbouring cell. The final outcome is an orientation PDF, which could be interpreted as the global optical flow of the scene.

4.2 Path Discovery

The work described in the previous sections provides two PDFs: one for the occurrence and one for the orientation of a scene. To discover the main paths, the information and the extracts of those corresponding to higher likelihood/probability need to be combined. Ideally the paths are identified corresponding to the modes of a probability density function that combines both occurrence and orientation information.

In order to estimate the main paths make a number of assumptions was made.

Path origin: The assumption is that all paths originate from the boundaries of the scene. Consequently, path discovery is started from a cell at the boundary of the scene and having high occurrence probability. This assumption would not work if the scene had an entrance or exit in the middle of the image, but this can be overcome relatively easily by using user-defined boundaries.

Graceful continuation/Smooth trajectory: As observed, the paths have a high probability to maintain their orientation (e.g. people are more likely to go on a straight line, and seldom go backwards.) So the expected direction of motion is modelled with a Poisson distribution, with its maximum in the neighboring cell along the current direction of motion.

The idea is to spread the likelihood of change in direction unevenly, maintaining the previous orientation as the one with highest probability, and forcing the other directions (change in direction) to have a lower likelihood. From the start point, the probability is calculated for each neighboring block using the occurrence PDF (PDFocc), the block matching accumulator (P_b) and the orientation probability (PDF_{or}). Furthermore, to avoid repeating calculations from the same block, the visited cells are marked and their probability is set to to 0 each time the path discovery process has to deal with them.

The process will follow the highest probability block. Also, a way of deciding when to split a trajectory in two or more sub-trajectories is devised. This technique works on a threshold that estimates whether two or more paths are viable, given their associated likelihood. However, so as not to generate too many branches, only a single split along a trajectory is admitted.

Once all paths are identified, a fitting process takes place. This serves two purposes: (i) to have a compact representation of the path, (ii) to have a faster way of estimating the distance between a blob/bounding rectangle, identified by new foreground data, and the spline, and consequently estimating an error. The final path is represented as a curve by fitting a uniform Cubic B-spline.

4.3 Self-Organizing Map for Learning Crowd Dynamics

The previous approach is based on background modelling, which cannot work properly under extremely crowded situations. The crowd PDFs derived by the described method are not global statistics. Also, the number of dimensions of the model is relatively high, especially for the orientation PDF. Those are disadvantages can be overcome by the method described in this section.

Here we describe some work carried out applying self-organized maps to learn the dominant crowd dynamics. The self-organized map (SOM) model [32] is a well known dimensionality reduction method proved to bear resemblance with some features of the human brain, which represent different sensory input by topologically ordered computational maps. SOMs are widely used in mapping multidimensional data onto a low-dimensional map. Examples of applications include the analysis of banking data, linguistic data [50] and image classification [55]. This section proposes a system learning the crowd dynamics with the SOM. The system uses dynamics information as input; and it generates SOM which captures the dominant recurrent dynamics.

4.3.1 Building SOM for a Crowded Scene

The most common SOMs have neurons organized as nodes in a one- or two-dimensional lattice. The neurons of a SOM are activated by input patterns in the course of a competitive learning process. At any moment in time only one output neuron is active, the so called winning neuron. Input patterns are from a n-dimensional input space and are then mapped to the one- or two- dimensional output space of the SOM. Every neuron has a weight vector which belongs to the input space.

The desirable SOM in this application should capture the two major components of the crowd dynamics: occurrence and orientation. Thus, a four dimensional input space is chosen to be the weight space of the SOM, which can be represented as $f:(x,y,\theta,\rho)$. Each data from the input space can be explained as the location where crowd moves and the motion vectors in the form of angle (θ) and magnitude (ρ). The SOM used in this experiment is organized in a two-dimensional space and represented by a square lattice.

There are two phases for tuning the SOM with an input pattern I, competing and updating. In the competing phase every neuron is compared with I; the similarly of I and the weights of all of the neurons are computed; and the neuron $N(i_w, j_w)$(denoted by the neuron's coordinates of the lattice) with highest similarity is selected as the winning neuron. In the update phase, for each neuron $N(i,j)$, a distance is calculated as:

$$d^2 = (i - i_w)^2 + (j - j_w)^2 \tag{9}$$

the topological neighborhood function is then defined as:

$$h(n) = \exp(-\frac{d^2}{2\sigma^2(n)}) \tag{10}$$

where n denotes the time, which can also be explained as the number of iterations. and $\sigma^2(n)$ decreases with the time. The weight of each neuron $N(i,j)$ at time $n+1$ is then defined by:

$$w(n+1) = w(n) + \eta(n)h(n)(x - w(n)) \tag{11}$$

where $w(n)$ and w(n+1) is the weight of the neuron at time n and $n+1$. $\eta(n)$ is the function of learning rate, which always decreases with time.

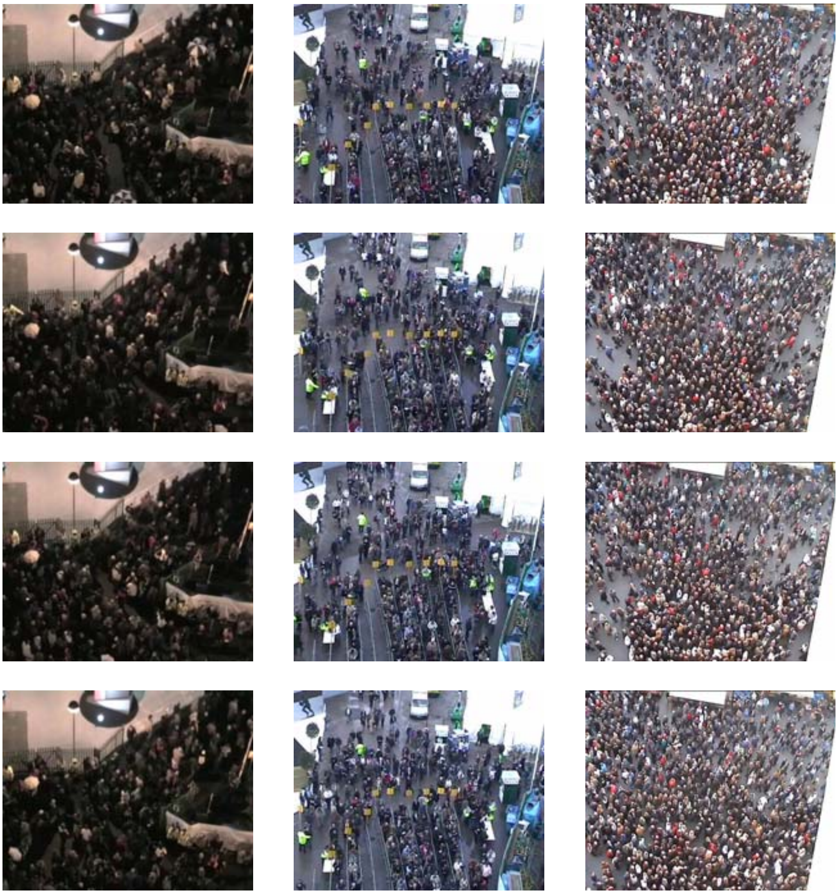


Fig. 10 The example frames from three different scenes

4.3.2 Visualization

Figure 10 illustrates three different video sequences with different dynamics. These video sequences have been input into the system, and Figure 11 shows the output SOMs. In the figure SOMs are visualized in the input space, i.e. showing the weight vector of each neuron. In the visualization, the colour arrows and their locations are from the weight vector of neurons; the location of the arrows are from the first two components of the weight vectors (x,y), and the arrows show the second two components - the components of motion (θ,ρ). The different colours of the arrows are also indicating the different orientation of the motion.

In the first video (the left column in Figure 10) the major crowd is moving from bottom left to top right of the scene. There is another crowd flow from bottom right of the scene which joins the major flow. In its SOM (the first one in Figure 11) the neurons with green arrows are clearly from the major flow and the ones with

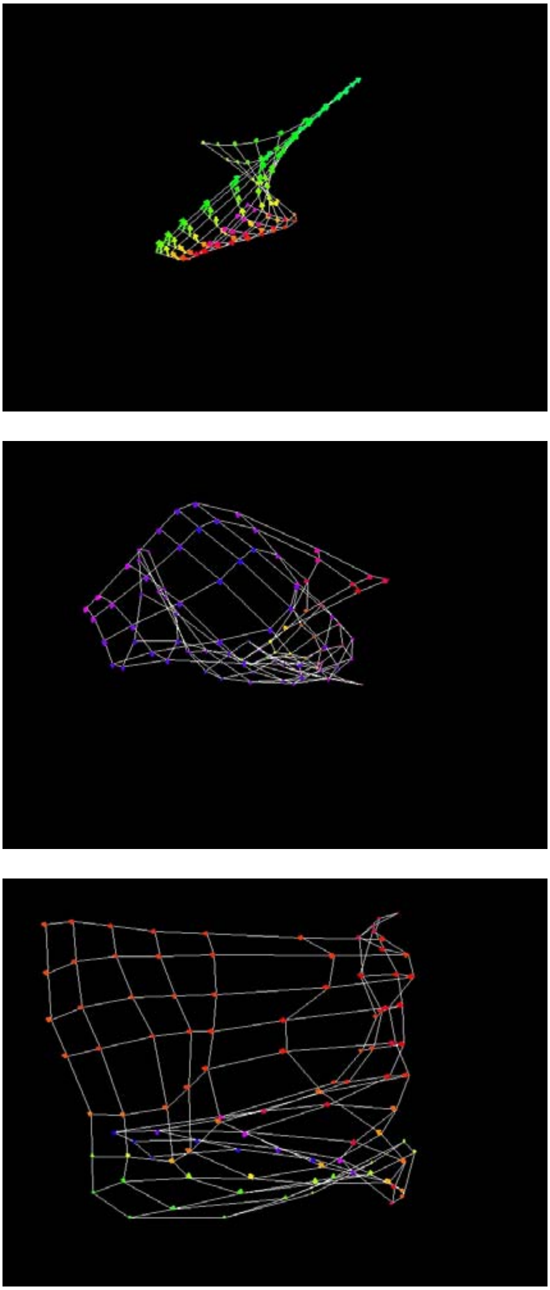


Fig. 11 The visualization of built SOMs

red and purple arrows are from the minor flow. In the second video (the middle column in Figure 10 it is an area of an entrance to a public space. So most of the people move from top to bottom of the scene. The crowd in the upper part of the scene is sparser and moves faster when compared to the crowd in the lower part of the scene. There is also a minor flow, which joins the major flow from the right of the scene. In the built SOM (the second SOM in Figure 11), again the flows are clearly indicated. Furthermore, the SOM takes an umbrella shape, which represents the shape of the flow constrained by the obstacles in the scene. In the third video (the right column in Figure 10) the scene is a large open area with multiple crowd flows. The major flow is moving from right to left; however there are several minor flows, most of which are in the lower part of the scene. Again the SOM (the third in Figure 11) captures the major dynamics and also some minor flows. From the three examples, it can be concluded that the SOMs not only preserves the dominant motion vector, but also represents the shape of the regions with dominant motion of the scenes.

5 Discussion

This chapter has described novel methods for an intelligent system which can automatically analyze crowd phenomena. The methods are based on computer vision techniques, combined with statistical techniques and a neural network. In particular, local descriptors matching with refined constraints are proposed to tackle the problem of crowd motion measurements. Two novel algorithms to estimate the motion of a crowd in complex scenes are presented, evaluated and compared in this chapter. The first algorithm employs Harris corner points and topological constraints are applied to make the matching of the points more robust. The second algorithm makes use of shape information. Local maximum curvatures are used as local descriptors and the edgelet constraints are enforced for the refined matching.

Statistical methods using Probability Density Functions are employed to learn the crowd dynamics by mining the main path of the crowded scene. Two PDFs (PDF_{occ} and PDF_{or}) are generated during this process. A path recovering method is developed by calculating the probability along the path using the PDFs. The results show that this work is a simple approach with reasonable results. Another approach of crowd dynamics learning adapts Self Organizing Maps to capture the main recurrent dynamics. There are a couple of possible extensions of the work. Especially for latter approach, analyzing the organization of the SOM would make it possible to understand the characters of the dynamics. Also the development of a metric for comparing SOMs could be very useful to enhance the automatic classification of crowded scenes.

Acknowledgements. This work was partially supported by the British Telecom Group PLC.

References

1. Adang, O.M., Stott, C.: A European study of the interaction between police and crowds of foreign nationals considered to pose a risk to public order, http://policestudies.homestead.com/Euro2004.html
2. ADVISOR: http://advisor.matrasi-tls.fr/
3. AEA, Techology: A technical summary of the aea egress code. Technical Report 1 (2002)
4. Andrade, E., Fisher, R.: Simulation of crowd problems for computer vision. In: First International Workshop on Crowd Simulation, vol. 3, pp. 71–80 (2005)
5. Andrade, E., Fisher, R.: Hidden Markov models for optical flow analysis in crowds. In: Proceedings of the 18th International Conference on Pattern Recognition (ICPR 2006), Washington, DC, USA, vol. 01, pp. 460–463. IEEE Computer Society, Los Alamitos (2006)
6. Andrade, E., Fisher, R.: Modelling crowd scenes for event detection. In: Proceedings of the 18th International Conference on Pattern Recognition (ICPR 2006), vol. 01, pp. 175–178. IEEE Computer Society, Washington (2006)
7. Andrade, E.L., Blunsden, S., Fisher, R.B.: Performance analysis of event detection models in crowded scenes. In: Proc. Workshop on Towards Robust Visual Surveillance Techniques and Systems at Visual Information Engineering 2006, Bangalore, India, pp. 427–432 (2006)
8. Antonini, G., Bierlaire, M., Weber, M.: Simulation of pedestrian behaviour using a discrete choice model calibrated on actual motion data. In: 4th STRC Swiss Transport Research Conference, Ascona (2004)
9. Antonini, G., Venegas, S., Thiran, J.P.: A discrete choice pedestrian behaviour model in visual tracking systems. In: Advanced Concepts for Intelligent Vision Systems, Brussels, Belgium, pp. 273–280 (2004)
10. Banarjee, S., Grosan, C., Abarha, A.: Emotional ant based modeling of crowd dynamics. In: Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2005), pp. 279–286 (2005)
11. Beauchemin, S., Barron, J.: The computation of optical flow. ACM Computing Surveys (CSUR) 27(3), 433–466 (1995)
12. BEHAVE: http://www.homepages.informatics.ed.ac.uk/rbf/BEHAVE/
13. Blackman, S.: Multiple hypothesis tracking for multiple target tracking. IEEE Aerospace and Electronic Systems Magazine 19(1), 5–18 (2004)
14. Boghossian, B., Velastin, S.: Motion-based machine vision techniques for the management of large crowds. In: The 6th IEEE International Conference on Electronics, Circuits and Systems, vol. 2 (1999)
15. Bouguet, J.: Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm. Intel Corporation, Microprocessor Research Labs (2000)
16. Brenner, M., Wijermans, N., Nussle, T., de Boer, B.: Simulating and controlling civilian crowds in robocup rescue. In: Proceedings of RoboCup 2005: Robot Soccer World Cup IX. Osaka (2005)
17. Brostow, G., Cipolla, R.: Unsupervised Bayesian Detection of Independent Motion in Crowds. In: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 594–601. IEEE Computer Society, Washington (2006)

18. Cai, Y., de Freitas, N., Little, J.J.: Robust visual tracking for multiple targets. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 107–118. Springer, Heidelberg (2006)
19. Chan, M.T., Hoogs, A., Bhotika, R., Perera, A., Schmiederer, J., Doretto, G.: Joint recognition of complex events and track matching. In: CVPR 2006: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1615–1622. IEEE Computer Society, Washington (2006) http://dx.doi.org/10.1109/CVPR.2006.160
20. Chang, T., Gong, S., Ong, E.: Tracking multiple people under occlusion using multiple cameras. In: British Machine Vision Conference, pp. 566–575 (2000)
21. Crowd, Dynamics: http://www.crowddynamics.com/
22. Crowd, MAGS: http://www2.ift.ulaval.ca/muscamags/Dnd-crowdmags-project.htm
23. Cupillard, F., Bremond, F., Thonnat, M.: Behaviour recognition for individuals, groups of people and crowd. IEE Seminar Digests 7 (2003)
24. Cupillard, F., Bremond, F., Thonnat, M., INRIA, F.: Group behavior recognition with multiple cameras. In: Sixth IEEE Workshop on Applications of Computer Vision, 2002 (WACV 2002). Proceedings, pp. 177–183 (2002)
25. Davies, A., Yin, J., Velastin, S.: Crowd monitoring using image processing. Electronics & Communication Engineering Journal 7(1), 37–47 (1995)
26. Dong, L., Parameswaran, V., Ramesh, V., Zoghlami, I.: Fast Crowd Segmentation Using Shape Indexing, Rio de Janeiro, Brazil (2007)
27. Doucet, A., Godsill, S., Andrieu, C.: On sequential Monte Carlo sampling methods for Bayesian filtering (2000)
28. Elgammal, A., Davis, L.: Probabilistic framework for segmenting people under occlusion. In: Eighth IEEE International Conference on Computer Vision, 2001. ICCV 2001. Proceedings, vol. 2, pp. 145–152 (2001)
29. Gabriel, P., Hayet, J., Piater, J., Verly, J.: Object tracking using color interest points. In: Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance, pp. 159–164 (2005)
30. Gabriel, P., Verly, J., Piater, J., Genon, A.: The state of the art in multiple object tracking under occlusion in video sequences. Advanced Concepts for Intelligent Vision Systems, 166–173 (2003)
31. Gouet, V., Boujemaa, N.: About optimal use of color points of interest for content-based image retrieval. Technical Report pp. RP–4439 (2002)
32. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River (1994)
33. Helbing, D., Farkas, I., Vicsek, T.: Simulating Dynamical Features of Escape Panic. Letters to Nature 407, 487–490 (2000)
34. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Physical Review E 51(5), 4282–4286 (1995)
35. Helbing, D., Molnar, P.: Self-organization phenomena in pedestrian crowds (1997), http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/9806152
36. Horn, B., Schunck, B.: Determining Optical Flow. Artificial Intelligence 17(1-3), 185–203 (1981)
37. Hu, W., Tan, T., Wang, L., Maybank, S.: A survey on visual surveillance of object motion and behaviors. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 34(3), 334–352 (2004)

38. Huang, C., Ai, H., Li, Y., Lao, S.: Vector boosting for rotation invariant multi-view face detection. In: Tenth IEEE International Conference on Computer Vision, vol. 1, pp. 446–453 (2005)
39. Hughes, R.: A continuum theory for the flow of pedestrians. Transportation Research Part B: Methodological 36(6), 507–535 (2002)
40. INRIA: http://www.inria.fr/rapportsactivite/RA2005/orion/uid1.html
41. Isard, M., Blake, A.: A mixed-state CONDENSATION tracker with automatic model-switching. In: IEEE International Conference on Computer Vision, pp. 107–112 (1998), http://citeseer.ist.psu.edu/isard98mixedstate.html
42. ISCAPS: http://www.iscaps.reading.ac.uk/home.htm
43. Jones, M., Viola, P.: Fast multi-view face detection. Mitsubishi Electric Research Lab TR-20003-96 (2003)
44. Kang, H., Kim, D., Bang, S.: Real-time multiple people tracking using competitive condensation. Proc. of the Intl. Conference on Pattern Recognition 1, 413–416 (2002)
45. Karlsson, R., Gustafsson, F.: Monte Carlo data association for multiple target tracking. Target Tracking: Algorithms and Applications (Ref. No. 2001/174), IEE 1 (2001)
46. Khan, S.M., Shah, M.: A multiview approach to tracking people in crowded scenes using a planar homography constraint. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 133–146. Springer, Heidelberg (2006)
47. Khan, Z., Balch, T., Dellaert, F.: MCMC-based particle filtering for tracking a variable number of interacting targets. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(11), 1805–1819 (2005)
48. Kim, K., Davis, L.S.: Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3953, pp. 98–109. Springer, Heidelberg (2006)
49. Kirchner, A., Schadschneider, A.: Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. Physica A: Statistical Mechanics and its Applications 312(1-2), 260–276 (2002)
50. Kirt, T., Vainik, E., Võhandu, L.: A method for comparing self-organizing maps: case studies of banking and linguistic data. In: Eleventh East-European Conference on Advances in Databases and Information Systems ADBIS, pp. 107–115. Technical University of Varna, Varna (2007)
51. Koller-Meier, E., Ade, F.: Tracking multiple objects using the Condensation algorithm. Robotics and Autonomous Systems 34(2-3), 93–105 (2001)
52. Kong, D., Gray, D., Tao, H.: Counting Pedestrians in Crowds Using Viewpoint Invariant Training. In: British Machine Vision Conference (2005)
53. Kong, D., Gray, D., Tao, H.: A viewpoint invariant approach for crowd counting. In: Proceedings of the 18th International Conference on Pattern Recognition (ICPR 2006), vol. 03, pp. 1187–1190 (2006)
54. Kretz, T., Schreckenberg, M.: F.a.s.t. - floor field- and agent-based simulation tool (2006)
55. Lefebvre, G., Laurent, C., Ros, J., Garcia, C.: Supervised Image Classification by SOM Activity Map Comparison. In: Proceedings of the 18th International Conference on Pattern Recognition (ICPR 2006), vol. 02, pp. 728–731 (2006)
56. Legion: http://www.legion.biz/about/index.html
57. Leibe, B., Seemann, E., Schiele, B.: Pedestrian detection in crowded scenes. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005, vol. 1 (2005)

58. Li, S.Z., Zhu, L., Zhang, Z., Blake, A., Zhang, H., Shum, H.: Statistical learning of multi-view face detection. In: Proceedings of the 7th European Conference on Computer Vision-Part IV, pp. 67–81. Springer, Heidelberg (2002)
59. Lin, S., Chen, J., Chao, H.: Estimation of number of people in crowded scenes using perspective transformation. IEEE Transactions on Systems, Man and Cybernetics, Part A 31(6), 645–654 (2001)
60. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. International Joint Conference on Artificial Intelligence 81, 674–679 (1981)
61. Ma, R., Li, L., Huang, W., Tian, Q.: On pixel count based crowd density estimation for visual surveillance. In: IEEE Conference on Cybernetics and Intelligent Systems, vol. 1 (2004)
62. Marana, A., da Costa, L., Lotufo, R., Velastin, S.: On the Efficacy of Texture Analysis for Crowd Monitoring. In: Proceedings of the International Symposium on Computer Graphics, Image Processing, and Vision, vol. 00, p. 354 (1998)
63. Marana, A., Da Fontoura Costa, L., Lotufo, R., Velastin, S.: Estimating crowd density with Minkowski fractal dimension. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, 1999. ICASSP 1999. Proceedings, vol. 6, pp. 3521–3524 (1999)
64. Marana, A., Velastin, S., Costa, L., Lotufo, R.: Estimation of crowd density using image processing. In: IEE Colloquium on Image Processing for Security Applications (Digest No: 1997/074), vol. 11 (1997)
65. Marana, A., Velastin, S., Costa, L., Lotufo, R.: Automatic estimation of crowd density using texture. Safety Science 28(3), 165–175 (1998)
66. Marques, J., Jorge, P., Abrantes, A., Lemos, J.: Tracking Groups of Pedestrians in Video Sequences. In: IEEE 2003 Conference on Computer Vision and Pattern Recognition Workshop, vol. 9, p. 101 (2003)
67. Mathes, T., Piater, J.: Robust non-rigid object tracking using point distribution models. In: Proc. of British Machine Vision Conference (BMVC), vol. 2 (2005)
68. Maurin, B., Masoud, O., Papanikolopoulos, N.: Monitoring crowded traffic scenes. In: The IEEE 5th International Conference on Intelligent Transportation Systems, 2002. Proceedings, pp. 19–24 (2002)
69. McKenna, S., Jabri, S., Duric, Z., Rosenfeld, A., Wechsler, H.: Tracking groups of people. Computer Vision and Image Understanding 80(1), 42–56 (2000)
70. Mittal, A., Davis, L.: M 2 Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene. International Journal of Computer Vision 51(3), 189–203 (2003)
71. Mokhtarian, F., Abbasi, S., Kittler, J.: Robust and efficient shape indexing through curvature scale space. In: Proc. British Machine Vision Conference, vol. 62 (1996)
72. Musse, S., Thalmann, D.: A Model of Human Crowd Behavior: Group Inter-Relationship and Collision Detection Analysis. In: Proc. Workshop of Computer Animation and Simulation of Eurographics, vol. 97, pp. 39–51 (1997)
73. Okuma, K., Taleghani, A., de Freitas, N., Little, J., Lowe, D.: A boosted particle filter: Multitarget detection and tracking. European Conference on Computer Vision 1, 28–39 (2004)
74. Pan, X., Han, C., Dauber, K., Law, K.: Human and social behavior in computational modeling and analysis of egress. Automation in Construction 15(4), 448–461 (2006)
75. PRISMATICA: `http://prismatica.king.ac.uk/`
76. Rahmalan, H., Nixon, M., Carter, J.: On Crowd Density Estimation for Surveillance. In: The Institution of Engineering and Technology Conference on Crime and Security, pp. 540–545 (2006)

77. Rasmussen, C., Hager, G.: Joint probabilistic techniques for tracking multi-part objects. In: 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1998. Proceedings, pp. 16–21 (1998)
78. Reid, D.: An algorithm for tracking multiple targets. IEEE Transactions on Automatic Control 24(6), 843–854 (1979)
79. Reisman, P., Mano, O., Avidan, S., Shashua, A., Ltd, M., Jerusalem, I.: Crowd detection in video sequences. In: IEEE 2004 Intelligent Vehicles Symposium, pp. 66–71 (2004)
80. Sidenbladh, H., Wirkander, S.: Tracking random sets of vehicles in terrain. In: Proc. 2003 IEEE Workshop on Multi-Object Tracking, vol. 9, p. 98 (2003)
81. Smith, K., Gatica-Perez, D., Odobez, J.: Using particles to track varying numbers of interacting people. In: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), vol. 1, pp. 962–969 (2005)
82. Sonka, M., Hlavac, V., Boyle, R.: Image Processing, Analysis, and Machine Vision. Tech. rep. (1998) ISBN 0-534-95393-X
83. Stauffer, C., Grimson, W.: Adaptive background mixture models for real-time tracking. In: 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (1999)
84. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
85. Swets, D., Punch, B.: Genetic algorithms for object localization in a complex scene. In: IEEE International Conference on Image Processing, pp. 595–598 (1995)
86. Stanford University: CIFE Seed Project 2004-2005, 2005-2006, http://eil.stanford.edu/egress/
87. Velastin, S., Yin, J., Davies, A., Vicencio-Silva, M., Allsop, R., Penn, A.: Automated measurement of crowd density and motion using imageprocessing. In: Seventh International Conference on Road Traffic Monitoring and Control, 1994, pp. 127–132 (1994)
88. Venegas, S., Knebel, S., Thiran, J.: Multi-object tracking using particle filter algorithm on the top-view plan. Technical report, LTS-REPORT-2004-003, EPFL (2004), http://infoscience.epfl.ch/getfile.py?mode=best&recid=87041
89. van Vliet, L., Young, I., beek, P.: Recursive gaussian derivative filters. In: Proc. l4th International Conference on Pattern Recognition (ICPR 1998), vol. 1, pp. 509–514. IEEE Computer Society Press, Los Alamitos (1998), http://citeseer.comp.nus.edu.sg/565386.html
90. Vu, V., Bremond, F., Thonnat, M.: Human Behaviour Visualisation and Simulation for Automatic Video Understanding. In: Proc. of the 10th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2002), Plzen–Bory, Czech Republic, pp. 485–492 (2002)
91. Wu, B., Nevatia, R.: Detection of Multiple, Partially Occluded Humans in a Single Image by Bayesian Combination of Edgelet Part Detectors. In: Tenth IEEE International Conference on Computer Vision, 2005. ICCV 2005, vol. 1, pp. 90–97 (2005)
92. Wu, B., Nevatia, R.: Tracking of multiple, partially occluded humans based on static body part detection. In: CVPR 2006: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 951–958 (2006)
93. Yin, J., Velastin, S., Davies, A.: Image Processing Techniques for Crowd Density Estimation Using a Reference Image. Proc. 2nd Asia-Pacific Conf. Comput. Vision 3, 6–10 (1995)
94. Zhan, B., Remagnino, P., Velastin, S.: Analysing Crowd Intelligence. In: Second AIxIA Workshop on Ambient Intelligence (2005)

95. Zhan, B., Remagnino, P., Velastin, S.: Mining paths of complex crowd scenes. In: Advances in Visual Computing: First International Symposium, pp. 126–133 (2005)
96. Zhan, B., Remagnino, P., Velastin, S.: Mining paths of complex crowd scenes. Lecture notes in computer science pp. 126–133 (2005), ISBN/ISSN 3-540-30750-8
97. Zhan, B., Remagnino, P., Velastin, S.: Visual analysis of crowded pedestrain scenes. In: XLIII Congresso Annuale AICA, pp. 549–555 (2005)
98. Zhan, B., Remagnino, P., Velastin, S., Bremond, F., Thonnat, M.: Matching gradient descriptors with topological constraints to characterise the crowd dynamics. In: IET International Conference on Visual Information Engineering, 2006. VIE 2006, pp. 441–446 (2006), ISSN: 0537-9989, ISBN: 978-0-86341-671-2
99. Zhan, B., Remagnino, P., Velastin, S., Monekosso, N., Xu, L.: A Quantitative Comparison of Two New Motion Estimation Algorithms. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Paragios, N., Tanveer, S.-M., Ju, T., Liu, Z., Coquillart, S., Cruz-Neira, C., Müller, T., Malzbender, T. (eds.) ISVC 2007, Part I. LNCS, vol. 4841, pp. 424–431. Springer, Heidelberg (2007)
100. Zhan, B., Remagnino, P., Velastin, S.A., Monekosso, N., Xu, L.Q.: Motion estimation with edge continuity constraint for crowd scene analysis. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Nefian, A., Meenakshisundaram, G., Pascucci, V., Zara, J., Molineros, J., Theisel, H., Malzbender, T. (eds.) ISVC 2006, Part II. LNCS, vol. 4292, pp. 861–869. Springer, Heidelberg (2006)
101. Zhao, T., Nevatia, R.: Tracking multiple humans in complex situations. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(9), 1208–1221 (2004)
102. Zhao, T., Nevatia, R.: Tracking multiple humans in crowded environment. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. II–406–II–413 (2004)

Part VI
Communications for CI Systems

Computational Intelligence for the Collaborative Identification of Distributed Systems

Giorgio Biagetti, Paolo Crippa, Francesco Gianfelici, and Claudio Turchetti

Abstract. In this chapter, on the basis of a rigorous mathematical formulation, a new algorithm for the identification of distributed systems by large scale collaborative sensor networks is suggested. The algorithm extends a KLT-based identification approach to a decentralized setting, using the distributed Karhunen-Loève transform (DKLT) recently proposed by Gastpar *et al.*. The proposed approach permits an arbitrarily accurate identification since it exploits both the asymptotic properties of convergence of DKLT and the universal approximation capabilities of radial basis functions neural networks. The effectiveness of the proposed approach is directly related to the reduction of total distortion in the compression performed by the single nodes of the sensor network, to the identification accuracy, as well as to the low computational complexity of the fusion algorithm performed by the fusion center to regulate the intelligent cooperation of the nodes. Some identification experiments, that have been carried out on systems whose behavior is described by partial differential equations in 2-D domains with random excitations, confirm the validity of this approach. It is worth noting the generality of the algorithm that can be applied in a wide range of applications without limitations on the type of physical phenomena, boundary conditions, sensor network used, and number of its nodes.

1 Introduction

In the last few years collaborative signal processing with distributed sources of data, signals, images and natural phenomena has been gaining importance.

Giorgio Biagetti, Paolo Crippa, Francesco Gianfelici, and Claudio Turchetti
DIBET – Dipartimento di Ingegneria Biomedica, Elettronica e Telecomunicazioni, Università Politecnica delle Marche, Via Brecce Bianche 12, I-60131 Ancona, Italy,
e-mail: g.biagetti@deit.univpm.it, p.crippa@univpm.it, f.gianfelici@deit.univpm.it, c.turchetti@univpm.it

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 475–500.
springerlink.com

Thanks to the recent advances in hardware technologies, today it is possible to realize low-power low-cost wireless devices with limited on-board processing capabilities and sensing units that are able to detect information from the distributed environment [1].

Even though individual sensors can only perform simple local computation and communicate over a short range at low data rate, when deployed in large numbers they can form an intelligent collaborative network interacting with the surrounding environment in a large spatial domain.

Sensor networks (SNs) characterized by low computational complexity, great learning capability, and efficient collaborative technology are highly desirable to discriminate, regulate and decide actions on real phenomena in many applications such as environmental monitoring, surveillance, factory instrumentation, defence and so on.

In classical multi-sensor schemes all the local sensors communicate their data to a central processor, that performs optimal estimation based on conventional statistical techniques. Conversely, in decentralized processing, some preliminary processing of data (often lossy compression) is locally carried out and condensed information is sent to a central processing unit, often known as the fusion center, so that the resulting sensor network has intelligence at each node [19]. Moreover, physical limitations can be present in the communication links between the sensors and the fusion center. In such cases, local data quantization/compression is not only a necessity, but also an integral part of the design of sensor networks. In addition, the sensor observations are usually corrupted by noise whose distribution can be difficult to characterize in practice, especially for large networks. Therefore, it is important to design optimal decentralized estimation schemes in the presence of unknown sensor noise as well as channel bandwidth limitations.

1.1 Sensor Networks: The State of the Art

Sensor networks represent an interesting and highly active research topic, with many important results being constantly discovered and published on the subject. In this section only a very concise summary of the main and most recent contributions is listed.

Kunniyur *et al.* determined in [27] that as the number of sensors grows there exists, under mild assumptions, a critical point in time determined solely by the lifetime distribution at which the number of emergent lacunae is asymptotically Poisson. Dana *et al.* established in [10]: *(i)* that for n nodes placed in a domain of fixed area, with probability converging to one as n grows, the power efficiency scales at least by a factor of $\sqrt{n}$, and *(ii)* a random network with n relay nodes and r simultaneous transmitter/receiver pairs located in a domain of fixed area can achieve a power efficiency that scales by a factor of $\sqrt{n}$. Dukes *et al.* derived in [13] that an arc-decomposition of the complete λ-fold directed graph $\overrightarrow{K}_n$ into directed complete bipartite subgraphs $\overrightarrow{K}_{a,b}$, as a model for ternary scheduling in wireless SNs, can be used to guarantee that every ordered pair of nodes can communicate in λ time slots. Baek *et al.* established in [2] that if the sensed data is bursty in space and time, then one can reap substantial benefits from aggregation and balancing. Georgiadis *et al.* determined in [20] that starting from a characterization of the optimal superflow amounts to obtaining a structural decomposition of the network in a sequence of disjoint subregions with decreasing overload, such that traffic flows only from regions of higher overload to regions of lower overload, it is possible to state that the optimal superflow represents the smoothest trajectory to overflow, followed by the network in case of instability. Toumpis *et al.* developed in [36] effective equations for the design of large wireless sensor networks that can be deployed in the most efficient manner, not only avoiding the formation of bottlenecks, but also striking the optimal balance between reducing congestion and having the data packets follow short routes. Dousse *et al.* determined in [12] that communications occurring at a fixed nonzero rate imply a fraction of the nodes to be disconnected. Gastpar *et al.* established in [18] that if all nodes act purely as relays for a single source-destination pair, then capacity grows with the logarithm of the number of nodes. Haenggi showed in [22] that the density function of the distance to the n-nearest neighbor of a homogeneous process in $\mathbb{R}^M$ is governed by a generalized gamma distribution. Barros *et al.* determined in [3] that the information as flow view provides an algorithmic interpretation for several results, among which perhaps the most important one is the optimality of implementing codes using a layered protocol stack. Chamberland *et al.* showed in [6] that on the basis of the Gartner-Ellis theorem and similar large-deviation theory results, it is possible to establish that performance improves monotonically with sensor density, whilst a finite sensor density is defined to be optimal in the stochastic signal case. Xiao *et al.* proposed in [41] a new decentralized estimation scheme that is universal in the sense that each sensor compression scheme requires only the knowledge of local SNR, rather than the noise probability distribution functions (pdf), while the final fusion step is also independent of the local noise pdfs. Franceschetti *et al.* determined in [15] that, on the basis of the Chen-Stein method of Poisson approximation, it is possible to show derivations and generalizations that are able to improve upon and simplify previous results that appeared in the literature. Xue *et al.* showed in [42]: *(i)* the exact threshold function for μ-coverage for wireless networks modeled as points uniformly distributed in a unit square, with every node connecting to its nearest neighbors, and *(ii)* that the network will be connected with probability approaching one. Yang *et al.* established in [43] that cooperative SNs with a mobile access point and no energy constraints but possibly misinformed nodes have the same capacity as with no misinformed sensors, if polling can be performed. Luo presented in [29] that: *(i)* the sensors are necessary and sufficient to jointly estimate the unknown parameter within a fixed root mean square error, and *(ii)* the optimal decentralized estimation scheme suggests allocating sensors to estimate the i-th bit. Liu *et al.* introduced in [28] an iterative algorithm to construct distributed quantizers that are person-by-person optimal. Tay *et al.* derived in [35] an asymptotically optimal strategy for the case where sensor decisions are only allowed to depend on locally available information, and global sharing of side information does not improve asymptotic performance, when the “Type I” error is constrained to be small.

1.2 Identification of Distributed Systems

The availability of scalable, flexibly deployable, and low-maintenance, energy-efficient sensor networks makes it possible to detect information on systems distributed in large domains in order to model their behavior. In mathematical language modeling a physical system corresponds to solve the system identification problem, namely given a set of input-output experimental data, find a mathematical model that describes the dynamic input-output relationship with sufficient accuracy. Although this problem has been extensively studied during the last four decades, and significant results have been obtained in this field [30], the general problem of identifying a distributed system on a large spatial domain has not been exhaustively investigated yet.

Several early works have demonstrated that one of the main features of neural networks such as Multilayer Perceptrons (MLPs) [9, 16, 26], Radial Basis Function (RBF) networks [31] and Approximate Identity Neural Networks (AINNs) [7, 37] is their ability to approximate some classes of input-output functions. These works have also shown how the degree of accuracy of the approximation depends on the learning algorithm as well as on the number of neurons available. Also, in the last years, neural networks have extensively used for modeling natural and artificial phenomena, for identifying and controlling dynamical systems as well as for approximating deterministic or random input–output functions representing unknown systems and/or their control laws.

Among these works, Schilling *et al.* in [34] presented an effective technique for approximating multivariate continuous functions with multidimensional raised-cosine type RBF networks. Ferrari *et al.* in [14] used an algebraic approach for approximating smooth multivariate nonlinear functions by feedforward neural networks. Huang *et al.* in [25] presented a new sequential learning algorithm for RBF networks, referred to as *generalized growing and pruning algorithm for RBF*, in order to approximate functions with any arbitrary distribution of input training samples/data. Huang *et al.* in [24] also proved in theory that the single-hidden-layer feedforward networks with randomly generated hidden nodes are actually universal approximators. Wu *et al.* in [40] proposed a novel neural organization of generalized adalines (adaptive linear elements) of Widrow for data driven function approximation. Wedge *et al.* in [39] presented an efficient function approximation through a hybrid RBF sigmoid neural network with a three-step training algorithm that utilizes both global search and gradient descent training. The algorithm used is intended to identify global features of an input-output relationship before adding local detail to the approximating function. Belli *et al.* in [4] showed that some classes of artificial neural networks exist such that they are capable of providing arbitrarily approximation, in the mean square sense, to prescribed stationary stochastic processes. Finally Crippa *et al.* in [8] and Turchetti *et al.* in [38] demonstrated the ability of stochastic neural networks to approximate nonlinear input-output random transformations, thus widening the range of applicability of these networks to nonlinear systems with memory.

Based on these premises, the aim of this chapter is to suggest an innovative framework for the identification of nonlinear non-stationary distributed systems. This approach is based on a centralized intelligent identifier that makes the best identification in a distributed setting on a chosen ensemble of realizations and with no constraints in terms of model kind and/or model order. Methodologically, we define a stochastic setting where the nonlinear system to be identified generates nondeterministic signals, i.e., stochastic processes (SPs), from given initial conditions and random parameters of input signals. In this way, the set of input-output pairs so obtained shows complex but identifiable geometrical relationships in the Hilbert spaces obtained by KLT-transformation of the outputs, because of the intrinsic separability property of the Karhunen-Loève transform (KLT) and uniformity properties of the systems considered. As a subsequent step we define a computational intelligence technique for approximating with arbitrary accuracy the previously mentioned mappings that are able to globally identify the distributed system.

The global optimization of the identification performance, the computational complexity, and the interactions between network elements is performed in a collaborative setting, exploiting and developing the cooperation mechanisms that underpin other related methodologies such as the distributed KLT [17]. This approach is particularly suitable in this context since an iterative algorithm that cooperatively minimizes the overall compression-induced distortion is suggested, and its convergence properties stated and proved.

The collaborative optimization of the distributed identification is a key factor for the extension from a centralized setting, in fact it allows to manage the evolution of the identifier with respect to the phenomena under analysis, the devolution of nodes, and the cooperative diversity of the network nodes, and all the other aspects that characterize sensor networks.

The proposed technique has been proven on physical models of real phenomena described by partial differential equations (PDEs) with random input signals and simulated in a CAD/CAM simulation environment. The experimental results obtained clearly show the effectiveness of the suggested identification methodology and its excellent compression capability. With no limitations on the type of model, environment geometry and model order the technique represents an innovative and very powerful framework in a large number of applications.

This chapter is organized as follows. Section 1 introduces the state of the art and the current best practices of sensor networks and *ad-hoc* networks. In Section 2 the mathematical representation of distributed systems using the KLT is given. Section 3 is devoted to the KLT-based algorithm, defined by means of an approximating nonlinear mapping based on RBF networks, for the identification of distributed systems. In Section 4 the problem of identifying a distributed system by means of a network of independent sensors is faced and solved. Section 5 reports the experimental results achieved identifying some physical systems whose behaviors are described by partial differential equations. Finally Section 6 concludes this chapter.

2 Modeling a Distributed System by the Karhunen-Loève Transform

Let us consider in a domain $\mathcal{D} \subset \mathbb{R}^Z$ the functional $\mathcal{F}$, depending on the position point $\mathrm{p} = \left[p' \; p'' \; \ldots \; p^{(Z)}\right]^T \in \mathcal{D}$, the time t, a generic input signal $u(t)$ and some initial condition c,

$$y\left(t, \mathrm{p}\right) = \mathcal{F}\left(u\left(\tau\right)\big|_{[t_0,t)}, t, \mathrm{p}, \mathrm{c}\right), \quad y \in \mathbb{R} \tag{1}$$

represents a scalar field, that is the behavior of a distributed system to be identified.

Well known examples of distributed systems are those described by partial differential equations.

The main problem in system identification is due to the great extent of input space which makes it difficult to achieve an exhaustive stimulation of the system. Nevertheless, in most application problems a complete description of the system is not required, since it is sufficient to restrict the input space to the subset $\mathcal{U}$ of signals actually occurring in the problem under observation. The identification of systems can thus be more effective if the input and output signals u and y are regarded as stochastic processes (SPs) $\boldsymbol{u}$ and $\boldsymbol{y}$, and the vector c of the initial conditions is also reckoned as a random variable (RV) vector $\mathbf{c}$.

Let us introduce the set $\mathcal{H}$ of all real-valued random variables (RVs) $\mathbf{a}$, satisfying the relationships $E\{\mathbf{a}\} = 0$ and $E\left\{|\mathbf{a}|^2\right\} < \infty$ where the symbol $E\{\cdot\}$ denotes the expectation. With the above considerations in mind, we assume that $\mathcal{U}$ is spanned by the signals $\boldsymbol{u}(t, \mathbf{a})$ where $\mathbf{a}$ is an RV that parameterizes the process $\boldsymbol{u}(t)$. Correspondingly, $\boldsymbol{y}(t, \mathbf{a})$ varies within a subset $\mathcal{Y}$ so that (1) establishes a transformation between SPs given by

$$\boldsymbol{y}(t) = \boldsymbol{y}(t, \mathrm{p}, \mathbf{x}) = \mathcal{F}\left\{\boldsymbol{u}(\tau, \mathbf{a})|_{[t_0,t)}, t, \mathrm{p}, \mathbf{c}\right\}, \quad u \in \mathcal{U},\ y \in \mathcal{Y} \tag{2}$$

where $\mathbf{x} = [\mathbf{a}\,\mathbf{c}^T]^\mathrm{T}$ is the collection of all parameters $\mathbf{a}$ related to the input process and the initial conditions $\mathbf{c}$, so that the system output is a function of the time t and vector $\mathbf{x}$ alone. This assumption is equivalent to restricting the input space only to the most likely signals.

Assuming the domain $\mathcal{D}$ has been discretized by a mesh defined by the nodal points $\mathrm{p}_1, \mathrm{p}_2, \ldots, \mathrm{p}_S$ then (1) reduces to S functionals

$$\boldsymbol{y}_\ell\left(t, \mathbf{x}\right) = \boldsymbol{y}\left(t, \mathrm{p}_\ell, \mathbf{x}\right) = \mathcal{F}\left(u\left(\tau, \mathbf{a}\right)\big|_{[t_0,t)}, t, \mathrm{p}_\ell, \mathbf{c}\right), \quad \ell = 1, \ldots, S \tag{3}$$

Moreover time discretization yields to

$$\boldsymbol{y}_\ell\left(n, \mathbf{x}\right) = \boldsymbol{y}\left(n, \mathrm{p}_\ell, \mathbf{x}\right) = \mathcal{F}\left(u\left(\eta, \mathbf{a}\right)\big|_{[0,n)}, n, \mathrm{p}_\ell, \mathbf{c}\right) \\ \ell = 1, \ldots, S\,, \quad n = 0, 1, \ldots, L_s - 1 \tag{4}$$

Let us introduce the vectors $\mathbf{y}_\ell \in \mathbb{R}^{L_s \times 1}$

$$\mathbf{y}_\ell \triangleq [\boldsymbol{y}_\ell\,(0)\ \cdots\ \boldsymbol{y}_\ell\,(L_s-1)]^T\,, \qquad \ell = 1, \ldots, S \tag{5}$$

and assume $L = S\ L_s$, the vector $\mathbf{y} \in \mathbb{R}^{L\times 1}$,

$$\begin{aligned}\mathbf{y} &\triangleq \left[\mathbf{y}_1^T\,\mathbf{y}_2^T\ \cdots\ \mathbf{y}_S^T\right]^T \\ &= [\boldsymbol{y}_1\,(0)\ \boldsymbol{y}_1\,(1)\ \cdots\ \boldsymbol{y}_1\,(L_s-1)\ \cdots\ \boldsymbol{y}_S\,(0)\ \boldsymbol{y}_S\,(1)\ \ldots\ \boldsymbol{y}_S\,(L_s-1)]^T\end{aligned} \tag{6}$$

is the discrete-space discrete-time representation of the scalar field $\boldsymbol{y}\,(t, \mathrm{p})$. This representation holds for every system that possesses the properties of uniformity and causality.

It is well-known that if $\mathcal{Y}$ is in a Hilbert space then $\mathbf{y}$, defined by (6) and with its realizations $\mathrm{y} \in \mathbb{R}^{L\times 1}$, can be represented by the Discrete Karhunen-Loève Transform (DKLT) [11], also called *canonical representation*. The DKLT and its inverse can be written in matrix form as

$$\begin{cases} \mathbf{y} = \Phi\, \mathrm{k}(\mathbf{x}) \\ \mathrm{k}(\mathbf{x}) = \Phi^T\, \mathbf{y} \end{cases} \tag{7}$$

where $\mathrm{k}(\mathbf{x}) \in \mathbb{R}^{V\times 1}$ is defined as $[\mathrm{k}(\mathbf{x})]_j = k_j(\mathbf{x})$, with $j = 1, \ldots, V$ and $V \leq L$. It is worth noting that the DKLT is the most efficient representation of the SP if the expansion is truncated to use fewer than L orthonormal basis vectors. The matrix $\Phi = [\phi_1\ \ldots\ \phi_V] \in \mathbb{R}^{L\times V}$ is the reduced orthogonal matrix whose columns ϕ_j, $j = 1, \ldots, V$, are the eigenvectors as obtained from the eigenvalue equation

$$\mathrm{R}_{\mathbf{yy}}\Phi = \Phi\Gamma \tag{8}$$

where $\mathrm{R}_{\mathbf{yy}} \in \mathbb{R}^{L\times L}$ is the autocorrelation matrix of $\mathbf{y}$ that is estimated as

$$\mathrm{R}_{\mathbf{yy}} = E\{\mathbf{y}\mathbf{y}^{\mathrm{T}}\} \tag{9}$$

and $\Gamma \in \mathbb{R}^{V\times V}$ is the diagonal matrix with (non-null) eigenvalues on the diagonal.

The main benefit of this representation is related to the separation property of KLT. On the basis of this property the output of the system can be expressed as a linear combination of products of a function of $\mathbf{x}$ alone and a function of n alone. Since the vectors ϕ_j are determined by means of $\mathrm{R}_{\mathbf{yy}}$, which can be estimated by the realizations of $\mathbf{y}$, the system identification reduces to modelling the functions $k_j(\mathbf{x})$, $j = 1, \ldots, V$. As $\mathbf{y}$ is a function of $\mathbf{x}$, the terms $k_j(\mathbf{x})$ describe on the space spanned by the columns of Φ curves $\mathcal{C}^j_{\mathbf{y}}(\mathbf{x})$, which all together characterize the SP $\mathbf{y}$. As an example, Fig. 1 displays the curves $\mathcal{C}^j_{\mathbf{y}}(\mathbf{x})$ described by the components $k_j(\mathbf{x})$, for $j = 1, \ldots, 6$ of a Duffing ordinary differential equation.

The properties of uniformity and causality determine a smooth behavior of these curves that have then to be reconstructed from an *ensemble* of points extracted by the described approach to perform the identification. Since $\mathrm{k}(\mathbf{x})$ is a no-memory input/output mapping, it can be approximated by a given vector function,

$$\mathrm{k}(\mathbf{x}) \approx \mathcal{G}[\mathbf{x}, \mathrm{W}] \tag{10}$$

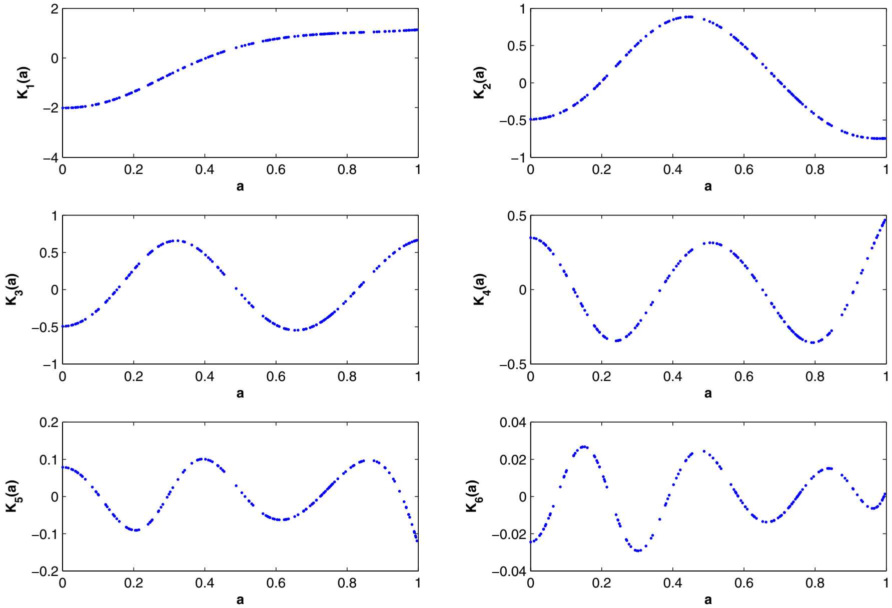


Fig. 1 The curves $\mathcal{C}^j_{\mathbf{y}}(\mathbf{x})$ described by the components $k_j(\mathbf{x})$, for $j = 1, \ldots, 6$ of a Duffing ordinary differential equation

where $\mathcal{G}[\cdot]$ is a nonlinear operator and $\mathrm{W} \in \mathbb{R}^{V \times M}$ is a matrix of parameters to be estimated.

3 Identification of a Distributed System Knowing the Output y

With the above considerations in mind, it can be stated that once the structure of the functional $\mathcal{G}[\mathbf{x}, \mathrm{W}]$ has been defined, the identification of the nonlinear system is equivalent to the estimation of the matrix W from an *ensemble* of the system's input-output pairs.

In order to derive the identification algorithm, it is necessary to relate the stochastic properties of the system (that allowed the development of the general theory) to the available *ensemble* of realizations. Let us then refer to these N realizations of $\mathbf{x}$ as $\mathrm{x}^{(i)} \in \mathbb{R}^{M_{\mathrm{x}} \times 1}$, with $i = 1, \ldots, N$, and to the corresponding realizations of $\mathbf{y}$ as $\mathrm{y}^{(i)} \in \mathbb{R}^{L \times 1}$, with $i = 1, \ldots, N$. Both can be put in matrix form as $\mathrm{X} = [\mathrm{x}^{(1)}\ \mathrm{x}^{(2)} \cdots\ \mathrm{x}^{(N)}]$ and $\mathrm{Y} = [\mathrm{y}^{(1)}\ \mathrm{y}^{(2)} \cdots\ \mathrm{y}^{(N)}]$, where $\mathrm{X} \in \mathbb{R}^{M_{\mathrm{x}} \times N}$ and $\mathrm{Y} \in \mathbb{R}^{L \times N}$. A currently used estimation $\hat{\mathrm{R}}$ of the autocorrelation matrix is given by

$$\mathrm{R} \approx \hat{\mathrm{R}} = \frac{1}{N}\mathrm{Y}\mathrm{Y}^T \tag{11}$$

where $\mathrm{R} \in \mathbb{R}^{L \times L}$.

The spectral representation of R is

$$\hat{\mathrm{R}}\,\mathrm{U} = \mathrm{U}\,\Lambda \tag{12}$$

where $\mathrm{U} = [\mathrm{u}_1\ \mathrm{u}_2\ \cdots\ \mathrm{u}_V] \in \mathbb{R}^{L\times V}$ is the matrix of eigenvectors and $\Lambda \in \mathbb{R}^{V\times V}$ the matrix of eigenvalues. By projecting all the N realizations onto the basis U we obtain the KLT representation

$$\begin{cases} \mathrm{y}^{(i)} = \mathrm{U}\mathrm{k}^{(i)} \\ \mathrm{k}^{(i)} = \mathrm{U}^T\mathrm{y}^{(i)} \end{cases} \qquad i = 1, \ldots, N \tag{13}$$

and, in matrix form,

$$\mathrm{K} = \mathrm{U}^T\,\mathrm{Y} \tag{14}$$

where $\mathrm{K} = [\mathrm{k}^{(1)}\ \mathrm{k}^{(2)} \cdots\ \mathrm{k}^{(N)}] \in \mathbb{R}^{V\times N}$. Once these projections have been obtained, the problem of approximating $\mathrm{k}(\mathbf{x})$ by a given function $\mathcal{G}[\mathbf{x}, \mathrm{W}]$ corresponds to finding the parameters W that make the following approximation

$$\mathrm{k}^{(i)} \approx \mathcal{G}[\mathrm{x}^{(i)}, \mathrm{W}], \qquad i = 1, \ldots, N \tag{15}$$

hold, so that a model of the system output is

$$\mathbf{y} \approx \mathrm{U}\,\mathcal{G}[\mathbf{x}, \mathrm{W}] \tag{16}$$

The estimation algorithm is particularly simple if $\mathcal{G}[\mathbf{x}, \mathrm{W}]$ is a linear function of W, namely

$$\mathcal{G}[\mathbf{x}, \mathrm{W}] = \mathrm{W}\,\mathrm{g}(\mathbf{x}), \tag{17}$$

with $\mathrm{g}(\mathbf{x})$ being an M-dimensional vector of suitable functions. By sampling these functions in correspondence of the N realizations of the parameterized input values and initial conditions vector, $\mathrm{x}^{(i)}$, $i = 1, \ldots, N$, we obtain the matrix $\mathrm{G} \in \mathbb{R}^{M\times N}$ defined as

$$\mathrm{G} = [\mathrm{g}(\mathrm{x}^{(1)})\ \mathrm{g}(\mathrm{x}^{(2)})\ \cdots\ \mathrm{g}(\mathrm{x}^{(N)})] \tag{18}$$

and the estimation problem reduces to estimating W so that

$$\mathrm{K} \approx \mathrm{W}\,\mathrm{G}\,. \tag{19}$$

3.1 Neural Network Based Identification

In general, being $\mathcal{G}[\mathbf{x}, \mathrm{W}]$ a nonlinear mapping, learning the proper weight matrix W is usually computationally very expensive. In this chapter we present an identification algorithm that is defined by means of an approximating *mapping* based on neural networks.

One of the main features of neural networks, allowing them to collect information from the environment, is their ability to learn by experience. Learning is a crucial activity for an intelligent system, whether artificial or biological, that aims at modeling the real world it interacts with. From a mathematical point of view, learning is related to the ability of the neural networks to approximate some classes of input-output functions. Several works have demonstrated that MLPs [9, 16, 26], RBF networks [31] and AINNs [7, 37] possess this property with reference to some classes of functions. These results show that neural networks of these kinds are capable of approximating arbitrarily well any function belonging to a certain class, the degree of accuracy depending on the learning algorithm as well as on the number of neurons available. Therefore, in the last years neural networks have gained much popularity in the modeling of natural and artificial phenomena, in the identification and control of dynamical systems as well as in the approximation of deterministic or random input–output functions representing unknown systems and/or their control laws [4, 34, 14, 25, 40, 24, 39, 33, 38].

In the following we will give a brief introduction to the RBF neural networks that have been considered as the *best choice* for our identifier. A radial basis function is a real-valued function $\gamma\left(\mathrm{x}\right)$ whose value depends only on the distance from a some point x_C, called a center, so that $\gamma\left(\mathrm{x},\mathrm{x}_C\right)=\gamma\left(\|\mathrm{x}-\mathrm{x}_C\|\right)$. The norm is usually the Euclidean distance. Radial basis functions are typically used to build up function approximations of the form

$$f\left(\mathrm{x}\right)=\sum_{i=1}^{M_\mathrm{n}}\mathrm{w}_i\,\gamma\left(\|\mathrm{x}-\mathrm{x}_{C_i}\|\right) \tag{20}$$

where the approximating function $f\left(\mathrm{x}\right)$ is represented as a sum of M_n radial basis functions, each associated with a different center x_{C_i}, and weighted by an appropriate coefficient w_i. The sum can be interpreted as a rather simple single-layer type of artificial neural network called a radial basis function network, with the radial basis functions taking on the role of the activation functions of the network. It can be shown that any continuous function on a compact interval can in principle be interpolated with arbitrary accuracy by a sum of this form, if a sufficiently large number M_n of radial basis functions are used. Girosi and Poggio [21, 32] have shown that RBF networks possess the property of *best approximation*. An approximation system has this property if, in the set of approximating functions (i.e. the set of functions corresponding to all possible choices of the adjustable parameters) there is one function which has minimum approximating error for any given function to be approximated. They also showed that this property is not shared by MLPs.

Being $\mathcal{G}[\mathbf{x},\mathrm{W}]$ a nonlinear mapping, learning the weight matrix W is usually computationally much more expensive than solving a linear problem such as (17). Thus, as a result of the considerations made above, the proposed nonlinear-in-the-parameter identifier is based on radial basis function networks [23, 5], so that the j-th component of the functional $\mathcal{G}[\mathbf{x},\mathrm{W}]$ defined in (10) can be put in the following form

$$[\mathcal{G}[\mathbf{x}, \mathrm{W}]]_j = \sum_{l=1}^{M_\mathrm{n}} [\omega_j]_l \exp\left(-[\chi_j]_l \sum_{m=1}^{M_\mathrm{x}} \left([\mathbf{x}]_m - [\Xi_j]_{l,m} \right)^2 \right) \quad , \qquad j = 1, \ldots, V \tag{21}$$

where M_n is the number of neurons in the RBF network, M_x is the dimension of vector $\mathbf{x}$, and $\omega_j \in \mathbb{R}^{M_\mathrm{n} \times 1}$, $\chi_j \in \mathbb{R}^{M_\mathrm{n} \times 1}$, and $\Xi_j = [\xi_j^1 \; \xi_j^2 \; \cdots \; \xi_j^{M_\mathrm{x}}] \in \mathbb{R}^{M_\mathrm{n} \times M_\mathrm{x}}$ with $\xi_j^m \in \mathbb{R}^{M_\mathrm{n} \times 1}$, for $m = 1, \ldots, M_\mathrm{x}$, are vectors or matrices of weights within $\mathrm{W} = [\mathrm{w}_1 \; \mathrm{w}_2 \; \cdots \; \mathrm{w}_V]^T \in \mathbb{R}^{V \times M}$, with $M = M_\mathrm{n}\,(M_\mathrm{x} + 2)$, defined so that $\mathrm{w}_j = [\omega_j^T \; \chi_j^T \; (\xi_j^1)^T \; (\xi_j^2)^T \; \cdots \; (\xi_j^{M_\mathrm{x}})^T]^T$. Despite its complexity the neural network-based approximations allow for great flexibility in the choice of the number of free parameters and scale gracefully when M_x increases, thus posing themselves as an interesting option in many circumstances.

4 Identification of a Distributed System by a Network of Independent Sensors

Let us consider the problem of identifying a distributed system by using a sensor network. In this case the generic variable $y_\ell\,(n)$, $\ell = 1, ..., S$, $n = 0, 1, ..., L_s - 1$ corresponds to the ℓ-th sensor and we assume the sensors are able to transmit only the observed subvector to a *fusion center* and cannot communicate to each other.

By applying a KLT to the observations of each sensor ignoring the dependencies with other terminals we obtain the *marginal KLT*, which is a particular case of (7) and is expressed as

$$\begin{bmatrix} \mathrm{k}_1(\mathbf{x}) \\ \mathrm{k}_2(\mathbf{x}) \\ \vdots \\ \mathrm{k}_S(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \Phi_{11} & 0 & \cdots & 0 \\ 0 & \Phi_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & \Phi_{SS} \end{bmatrix}^T \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_S \end{bmatrix} \tag{22}$$

with $\Phi_{ii} \in \mathbb{R}^{L_S \times V_i}$, $i = 1, \ldots, S$. As it is clearly stated by (22) the generic sensor ℓ transmits the subvector k_ℓ (or an approximation of it) so that the output vector $\mathbf{y}$ cannot be reconstructed with a negligible error. This means that the identification approach previously discussed cannot be applied directly to this case due to the lack of a complete knowledge of the output $\mathbf{y}$. It is easy to verify that the marginal KLT will lead to a suboptimal solution to this problem. In general we can search for a solution of the kind

$$\begin{bmatrix} \mathrm{h}_1(\mathbf{x}) \\ \mathrm{h}_2(\mathbf{x}) \\ \vdots \\ \mathrm{h}_S(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \Psi_1 & 0 & \cdots & 0 \\ 0 & \Psi_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & \Psi_S \end{bmatrix}^T \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_S \end{bmatrix} \tag{23}$$

with $\Psi_i \in \mathbb{R}^{L_S \times V_i}$, $\mathrm{h}_i \in \mathbb{R}^{V_i \times 1}$, $i = 1, \ldots, S$ or in a more compact form,

$$\mathrm{h}(\mathbf{x}) = \Psi^T \mathbf{y} \tag{24}$$

with $\Psi = \mathrm{diag}\left[\Psi_1, \ldots, \Psi_S\right] \in \mathbb{R}^{L \times V}$ and $\mathrm{h} \in \mathbb{R}^{V \times 1}$, $V = \sum_{i=1}^{S} V_i$. In this case the accuracy of the distributed identification is related both to the approximation of the mapping $\mathbf{k}\left(\mathbf{x}\right)$ and to the minimization of the error $E\left\{\|\mathbf{y} - \hat{\mathbf{y}}\|^2\right\}$ between the real system output $\mathbf{y}$ and its estimation $\hat{\mathbf{y}}$, based on the sensors' observations, and given by

$$\hat{\mathbf{y}} = \mathrm{R}\Psi\left(\Psi^T \mathrm{R}\Psi\right)^{-1} \mathrm{h}(\mathbf{x}) \; . \tag{25}$$

However in this case, to the best knowledge of authors, and as it is also pointed out in [17], it is not known a closed-form solution to this problem.

The algorithm developed by Gastpar *et al.*, also known as *distributed Karhunen-Loève transform* is an iterative procedure that aims at finding the matrix Ψ that achieves the MSE best estimate of $\hat{\mathbf{y}}$ in (25).

4.1 Two Sensors ($S = 2$)

In order to better illustrate the procedure, instead of using the general formulation (22), we start by considering the simple case of two sensors only, corresponding to the variables $\mathbf{y}_1$ and $\mathbf{y}_2$. Assuming the representation $\mathbf{y}_2$ given by the sensor S_2 to be fixed, we would determine the representation of $\mathbf{y}_1$ such that $E\left\{\|\mathbf{y} - \hat{\mathbf{y}}\|^2\right\}$ is minimum, $\hat{\mathbf{y}}$ being the approximation of $\mathbf{y}$.

The approximation provided by the second sensor can be expressed by $\mathrm{h}_2(\mathbf{x}) = \Psi_2^T \mathbf{y}_2 + \mathbf{z}_2$ where $\mathbf{z}_2$ are jointly Gaussian random variables independent of $\mathbf{y}_2$, with zero mean and covariance matrix R_z. Then we can partition the covariance matrix of the entire vector $\mathbf{y}$ into four parts, according to

$$\mathrm{R} = \begin{bmatrix} \mathrm{R}_{11} & \mathrm{R}_{12} \\ \mathrm{R}_{21} & \mathrm{R}_{22} \end{bmatrix} \tag{26}$$

where $\mathrm{R}_{ij} = E\left\{\mathbf{y}_i \mathbf{y}_j^T\right\}$ with $i, j = 1, 2$. Now to find the best estimate of $\mathbf{y}_1$ we define the matrix Ξ, as follows

$$\Xi = \left[\mathrm{R}_{12}^T \; \mathrm{R}_{22}^T \, \Psi_2\right] \begin{bmatrix} \mathrm{R}_{11} & \mathrm{R}_{12}\Psi_2 \\ \Psi_2^T \mathrm{R}_{12}^T & \Psi_2^T \mathrm{R}_{22} \Psi_2 + \mathrm{R}_z \end{bmatrix}^{-1} \tag{27}$$

Let Ξ^* consist of the first L_S column of Ξ, thus, we obtain a new matrix

$$\mathrm{R}_w = \begin{bmatrix} \mathrm{I}_{L_S} \\ \Xi^* \end{bmatrix} \left(\mathrm{R}_{11} - \mathrm{R}_{12}\Psi_2 \left(\Psi_2^T \mathrm{R}_{22} \Psi_2 + \mathrm{R}_z\right)^{-1} \Psi_2^T \mathrm{R}_{12}^T\right) \left[\mathrm{I}_{L_S} \;\; \Xi^{*\,T}\right] \tag{28}$$

with $\mathrm{R}_w \in \mathbb{R}^{L \times L}$, that can be reduced to the diagonal form

$$\mathrm{R}_w = \mathrm{D}_w \mathrm{diag}\left(\lambda_w^{(1)}, \ldots, \lambda_w^{(N)}\right) \mathrm{D}_w^T \tag{29}$$

with $\lambda_w^{(1)} \geq \lambda_w^{(2)} \geq \ldots \geq \lambda_w^{(N)}$, thus we can obtain the basis for representing y_1

$$\Psi_1^T = \left[\mathrm{D}_w^{(V_1)}\right]^T \begin{bmatrix} \mathrm{I}_{L_S} \\ \Xi^* \end{bmatrix} \tag{30}$$

where $\mathrm{D}_w^{(V_1)}$ denotes the matrix consisting of the first $V_1 \leq N$ columns of the matrix D_w and I_{L_S} is the L_S-dimensional identity matrix. In this way by truncating the representation to the first V_1 eigenvectors of $\mathrm{D}_w^{(V_1)}$ we obtain the parameters

$$\mathrm{h}_1(\mathbf{x}) = {\Psi_1}^T \mathbf{y}_1 \, . \tag{31}$$

4.2 S Sensors

Let us now consider the general case of S sensors, but in which all the variables $\mathbf{y}_1, \ldots, \mathbf{y}_S$ are known except for the j-th variable $\mathbf{y}_j$. Thus assuming the representation of $\mathbf{y}_1, \ldots, \mathbf{y}_{j-1}, \mathbf{y}_{j+1}, \ldots, \mathbf{y}_S$ given by the $S-1$ sensors to be fixed, we would determine the representation of $\mathbf{y}_j$ such that $E\left\{\|\mathbf{y} - \hat{\mathbf{y}}\|^2\right\}$ is minimum, $\hat{\mathbf{y}}$ being as usual the approximation of $\mathbf{y}$.

The approximation provided by the sensors with the fixed representation can be expressed as $\mathrm{k}_i(\mathbf{x}) = \Psi_i^T \mathbf{y}_i + \mathbf{z}_i$ with $i = 1, \ldots, S$ and $i \neq j$ where $\mathbf{z}_i$ are Gaussian random variables of zero mean and covariance matrix $\mathrm{R}_{z_i} \in \mathbb{R}^{V_i \times V_i}$, $(z_i \in \mathbb{R}^{V_i})$. Thus the covariance matrix $\mathrm{R} \in \mathbb{R}^{L \times L}$ of $\mathbf{y}$ can be written as

$$\mathrm{R} = \begin{bmatrix} \mathrm{R}_{11} & \mathrm{R}_{12} & \cdots & \mathrm{R}_{1S} \\ \mathrm{R}_{21} & \mathrm{R}_{22} & \cdots & \mathrm{R}_{2S} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{R}_{S1} & \mathrm{R}_{S2} & \cdots & \mathrm{R}_{SS} \end{bmatrix} \tag{32}$$

where $\mathrm{R}_{ik} = E\left\{\mathbf{y}_i \mathbf{y}_k^T\right\}$, $i, k = 1, \ldots, S$ and $\mathrm{R}_{ik} \in \mathbb{R}^{L_S \times L_S}$. Now let $\Xi_j \in \mathbb{R}^{L_S(S-1) \times V - V_j + L_S}$ be the matrix defined by

$$\Xi_j = \begin{bmatrix} \mathrm{R}_{11}\Psi_1 & \cdots & \mathrm{R}_{1j-1}\Psi_{j-1} & \mathrm{R}_{1j} & \mathrm{R}_{1j+1}\Psi_{j+1} & \cdots & \mathrm{R}_{1S}\Psi_S \\ \mathrm{R}_{21}\Psi_1 & \cdots & \mathrm{R}_{2j-1}\Psi_{j-1} & \mathrm{R}_{2j} & \mathrm{R}_{2j+1}\Psi_{j+1} & \cdots & \mathrm{R}_{2S}\Psi_S \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathrm{R}_{j-11}\Psi_1 & \cdots & \mathrm{R}_{j-1j-1}\Psi_{j-1} & \mathrm{R}_{j-1\,j} & \mathrm{R}_{j-1j+1}\Psi_{j+1} & \cdots & \mathrm{R}_{j-1S}\Psi_S \\ \mathrm{R}_{j+11}\Psi_1 & \cdots & \mathrm{R}_{j+1j-1}\Psi_{j-1} & \mathrm{R}_{j+1\,j} & \mathrm{R}_{j+1j+1}\Psi_{j+1} & \cdots & \mathrm{R}_{j+1S}\Psi_S \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathrm{R}_{S1}\Psi_1 & \cdots & \mathrm{R}_{Sj-1}\Psi_{j-1} & \mathrm{R}_{Sj} & \mathrm{R}_{Sj+1}\Psi_{j+1} & \cdots & \mathrm{R}_{SS}\Psi_S \end{bmatrix} .$$

$$\begin{bmatrix} \mathrm{Q}_{11}+\mathrm{R}_{z1} & \mathrm{Q}_{12} & \cdots & \mathrm{Q}_{1j-1} & \Psi_1^T\mathrm{R}_{1j} & \mathrm{Q}_{1j+1} & \cdots & \mathrm{Q}_{1S} \\ \mathrm{Q}_{21} & \mathrm{Q}_{22}+\mathrm{R}_{z2} & \cdots & \mathrm{Q}_{2j-1} & \Psi_2^T\mathrm{R}_{2j} & \mathrm{Q}_{2j+1} & \cdots & \mathrm{Q}_{2S} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathrm{Q}_{j-11} & \mathrm{Q}_{j-12} & \cdots & \mathrm{Q}_{j-1j-1}+\mathrm{R}_{zj-1} & \Psi_{j-1}^T\mathrm{R}_{j-1j} & \mathrm{Q}_{j-1j+1} & \cdots & \mathrm{Q}_{j-1S} \\ \mathrm{R}_{j1}\Psi_1 & \mathrm{R}_{j2}\Psi_2 & \cdots & \mathrm{R}_{jj-1}\Psi_{j-1} & \mathrm{R}_{jj} & \mathrm{R}_{jj+1}\Psi_{j+1} & \cdots & \mathrm{R}_{jS}\Psi_S \\ \mathrm{Q}_{j+11} & \mathrm{Q}_{j+12} & \cdots & \mathrm{Q}_{j+1j-1} & \Psi_{j+1}^T\mathrm{R}_{j+1j} & \mathrm{Q}_{j+1j+1}+\mathrm{R}_{zj+1} & \cdots & \mathrm{Q}_{j+1S} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathrm{Q}_{S1} & \mathrm{Q}_{S2} & \cdots & \mathrm{Q}_{Sj-1} & \Psi_S^T\mathrm{R}_{Sj} & \mathrm{Q}_{Sj+1} & \cdots & \mathrm{Q}_{SS}+\mathrm{R}_{zS} \end{bmatrix}^{-1} \tag{33}$$

where

$$\mathrm{Q}_{ik} = \Psi_i^T\mathrm{R}_{ik}\Psi_k \quad , \qquad i,k = 1,\ldots,S\;, \quad i,k \neq j \tag{34}$$

and $\mathrm{Q}_{ik} \in \mathbb{R}^{V_i \times V_k}$. Let the matrix $\Xi_j^* \in \mathbb{R}^{L_S(S-1)\times L_S}$ consist of the columns of Ξ_j from $\sum_{i=1}^{j-1} V_i + 1$ to $\sum_{i=1}^{j-1} V_i + L_S$. Then Ξ_j^* can be cast as

$$\Xi_j^* = \begin{bmatrix} \Xi_j'^* \\ \Xi_j''^* \end{bmatrix} \tag{35}$$

with $\Xi_j'^* \in \mathbb{R}^{L_S(S-1)(j-1)\times L_S}$ and $\Xi_j''^* \in \mathbb{R}^{L_S(S-1)(S-j)\times L_S}$, $j = 1,\ldots,S$. Let $\mathrm{I}_{L_S} \in \mathbb{R}^{L_S \times L_S}$ be the identity matrix, thus, we can obtain a new matrix

$$\mathrm{R}_{w,j} = \begin{bmatrix} \Xi_j'^* \\ \mathrm{I}_{L_S} \\ \Xi_j''^* \end{bmatrix} \left(\mathrm{R}_{jj} - \mathrm{C}_j\mathrm{H}_j^{-1}\mathrm{C}_j^T\right) \left[\, \Xi_j'^{*\,T} \;\; \mathrm{I}_{L_S} \;\; \Xi_j''^{*\,T} \,\right] \tag{36}$$

where

$$\mathrm{C}_j = \left[\, \mathrm{R}_{j1}\Psi_1 \;\; \mathrm{R}_{j2}\Psi_2 \cdots \mathrm{R}_{jj-1}\Psi_{j-1} \;\; \mathrm{R}_{jj+1}\Psi_{j+1} \cdots \mathrm{R}_{jS}\Psi_S \,\right] \tag{37}$$

and

$$\mathrm{H}_j = \begin{bmatrix} \mathrm{Q}_{11}+\mathrm{R}_{z1} & \mathrm{Q}_{12} & \cdots & \mathrm{Q}_{1j-1} & \mathrm{Q}_{1j+1} & \cdots & \mathrm{Q}_{1S} \\ \mathrm{Q}_{21} & \mathrm{Q}_{22}+\mathrm{R}_{z2} & \cdots & \mathrm{Q}_{2j-1} & \mathrm{Q}_{2j+1} & \cdots & \mathrm{Q}_{2S} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \mathrm{Q}_{j-11} & \mathrm{Q}_{j-12} & \cdots & \mathrm{Q}_{j-1j-1}+\mathrm{R}_{zj-1} & \mathrm{Q}_{j-1j+1} & \cdots & \mathrm{Q}_{j-1S} \\ \mathrm{Q}_{j+11} & \mathrm{Q}_{j+12} & \cdots & \mathrm{Q}_{j+1j-1} & \mathrm{Q}_{j+1j+1}+\mathrm{R}_{zj+1} & \cdots & \mathrm{Q}_{j+1S} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathrm{Q}_{S1} & \mathrm{Q}_{S2} & \cdots & \mathrm{Q}_{Sj-1} & \mathrm{Q}_{Sj+1} & \cdots & \mathrm{Q}_{SS}+\mathrm{R}_{zS} \end{bmatrix} \tag{38}$$

with $\mathrm{R}_{w,j} \in \mathbb{R}^{L \times L}$, $\mathrm{C}_j \in \mathbb{R}^{L_S \times V - V_j}$, and $\mathrm{H}_j \in \mathbb{R}^{V-V_j \times V-V_j}$. The matrix $\mathrm{R}_{w,j}$ can be reduced to the diagonal form

$$\mathrm{R}_{w,j} = \mathrm{D}_{w,j}\;\Lambda_j\;\mathrm{D}_{w,j}^T \tag{39}$$

where

Algorithm 1. Iterative refinement of sensor transform matrices.

```
Function Distributed_KLT(R)
{Algorithm initialization:}
for all $\ell \in \{1, \ldots, S\}$ do
    $\Psi_\ell \leftarrow \Phi_{\ell\ell}$
{Refinement:}
repeat
    for all $\ell \in \{1, \ldots, S\}$ do
        compute formulae (33)–(39)
        $\Psi_\ell \leftarrow$ result of (41)
until MSE does not improve any more
```

$$\Lambda_j = \mathrm{diag}\left(\lambda_{w,j}^{(1)}, \ldots, \lambda_{w,j}^{(N)}\right) \tag{40}$$

with $\lambda_{w,j}^{(1)} \geq \lambda_{w,j}^{(2)} \geq \ldots \geq \lambda_{w,j}^{(N)}$, and with $\mathrm{D}_{w,j} \in \mathbb{R}^{L \times N}$ and $\Lambda_j \in \mathbb{R}^{N \times N}$, thus we can obtain the representation for the j-th sensor

$$\Psi_j^T = \left[\mathrm{D}_{w,j}^{(V_j)}\right]^T \begin{bmatrix} \Xi_j'^{*} \\ \mathrm{I}_{L_S} \\ \Xi_j''^{*} \end{bmatrix} \tag{41}$$

where $\mathrm{D}_{w,j}^{(V_j)} \in \mathbb{R}^{L \times L_S}$ denotes the matrix consisting of the first V_j columns of the matrix $\mathrm{D}_{w,j}$. In this way by truncating the representation to the first V_j eigenvectors of $\mathrm{D}_{w,j}^{(V_j)}$ we obtain the parameters

$$\mathrm{h}_j(\mathbf{x}) = \Psi_j^T \mathbf{y}_j \ . \tag{42}$$

4.3 Best Estimate of the Matrix Ψ Based on the Distributed KLT Algorithm

On the basis of the previous formulation we are now in a position to establish a collaborative algorithm that progressively estimates the best approximation for a sensor as a function of the given representation for the other sensors. The pseudocode of the algorithm based on the Distributed KLT proposed by Gastpar *et al.* [17] is reported in Algorithm 1.

As it has been demonstrated in [17], the MSE decreases monotonically, i.e. $E\left\{\|\mathbf{y} - \hat{\mathbf{y}}^{(m)}\|^2\right\} \geq E\left\{\|\mathbf{y} - \hat{\mathbf{y}}^{(m+1)}\|^2\right\}$, where $\hat{\mathbf{y}}^{(m)}$ is the approximation obtained after having performed m iterations. This implies that the algorithm will converge to a stable point, a saddle point or a local minimum, but it clearly cannot guarantee the convergence to a global optimum.

Having obtained the best estimate of the matrices Ψ_ℓ, $\ell = 1, \ldots, S$, by the algorithm previously described and taking into account the results of Sect. 3, the identification of the distributed system, that is the model of the system, is represented by (25) where the function $\mathrm{h}(\mathbf{x})$ is approximated as

$$\mathrm{h}(\mathbf{x}) = \Psi^T \mathbf{y} \approx \mathcal{G}[\mathbf{x}, \mathrm{W}] \tag{43}$$

and $\mathcal{G}[\mathbf{x}, \mathrm{W}]$ is given by (21).

5 Experimental Results

To validate the identification technique suggested, the collaborative algorithm was applied to the identification of several distributed systems whose behavior can be described as the solution of a partial differential equation (PDE). These equations are used as a mathematical model in a wide variety of physical systems. As an example, elliptic and parabolic PDEs are used for modeling: *i)* steady and unsteady heat transfer in solids, *ii)* flows in porous media and diffusion problems, *iii)* electrostatics of dielectric and conductive media. At the same time hyperbolic PDEs are used for *i)* transient and harmonic wave propagation in acoustics and electromagnetics, *ii)* transverse motions of membranes.

In this section we consider two application examples regarding the solution of a parabolic and a hyperbolic PDE over simple geometries. The solutions of such equations were achieved by using the Matlab PDE-Toolbox. In this framework the equations are solved by the finite-element method with non uniform meshes.

5.1 First Experiment: Parabolic PDE

The term parabolic PDE is used for equations with spatial operators and first order time derivatives of the form:

$$d\frac{\partial y}{\partial t} - \nabla \cdot (c\nabla y) + ay = f \tag{44}$$

where y is the unknown, d is a complex valued function on a bounded domain $\mathcal{D}$ in the plane, and c, a, f are coefficients that can depend on time t, d can depend on time t as well.

The first experiment carried out made use of a scalar field obtained by the discretization of the following parabolic PDE, derived from (44) with $d = 1$, $c = 1$, $a = 5$, and $f = 10$:

$$\frac{\partial y}{\partial t} - \nabla^2 y + 5\,y = 10 \tag{45}$$

on the mesh shown in Fig. 2 where the elliptical equation domain $\mathcal{D}$ is reported. The excitation was given as a boundary condition $y = \sin(\omega\, t)$ applied to an arc of the boundary, with ω being proportional to the input parameter. We placed four sensors,

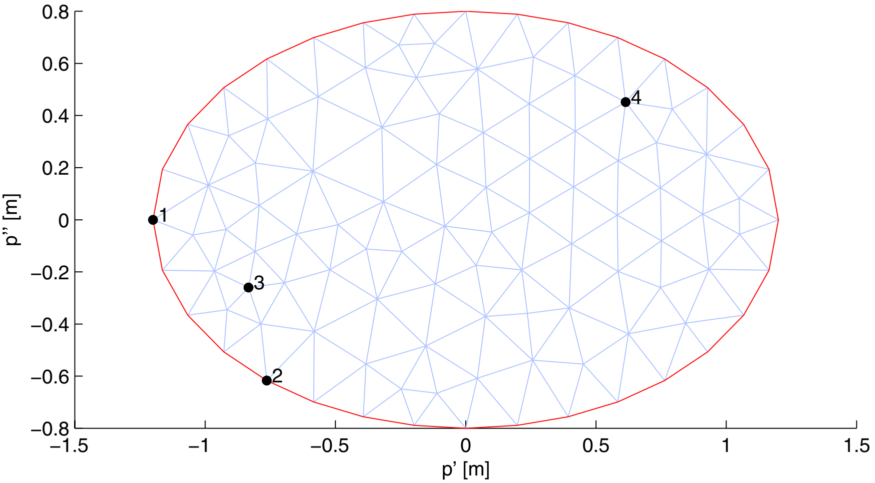


Fig. 2 Domain of the equation used in the first example and mesh used for its solution. Highlighted are the positions in which the sensors were placed

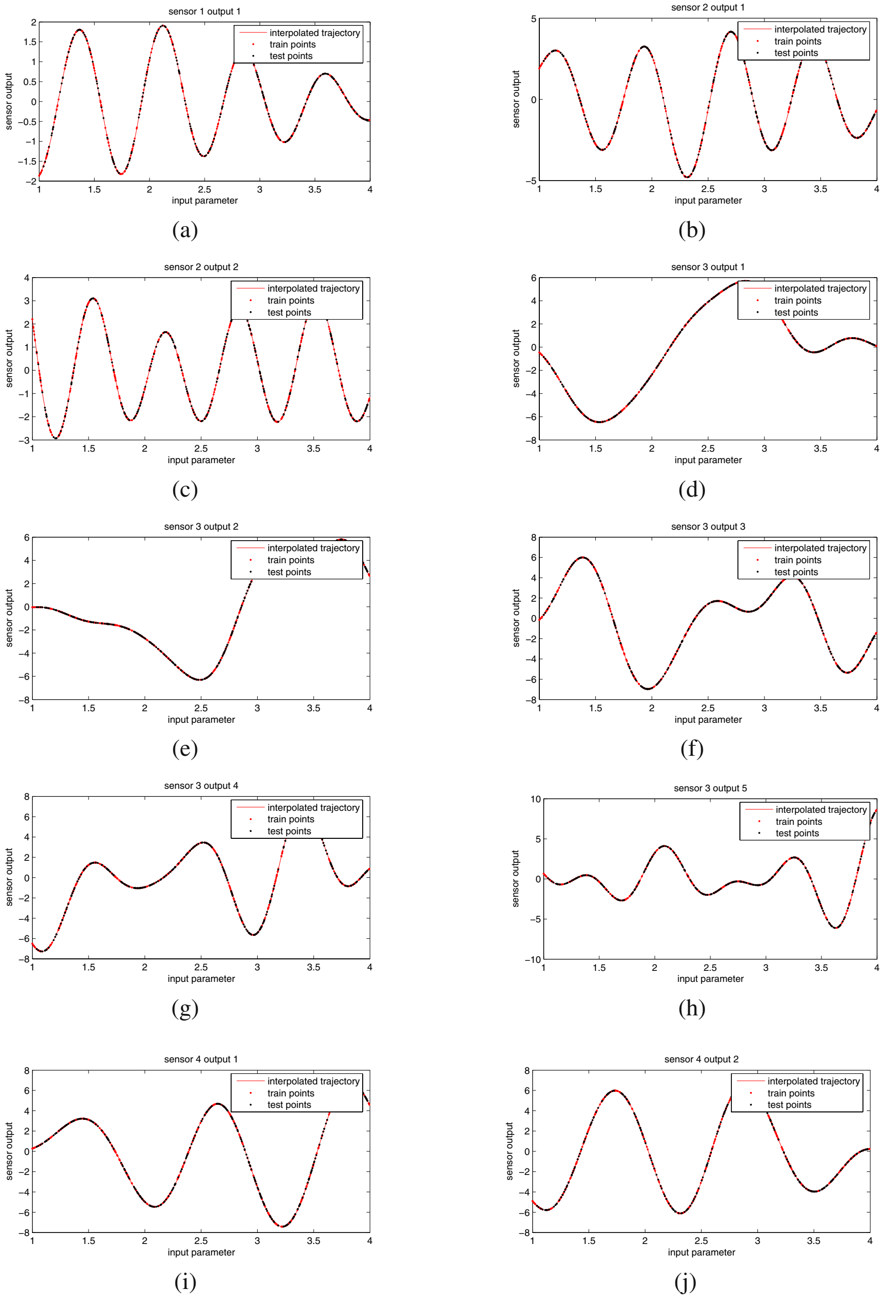
labeled 1–4, on randomly chosen knots and selected the best rate allocation for a fixed rate $V = 10$, which resulted to be $V_1 = 1$, $V_2 = 2$, $V_3 = 5$, $V_4 = 2$ i.e. the sensor 1 has one output, the sensor 2 has two outputs, the sensor 3 has five outputs and finally the sensor 4 has two outputs.

The results are shown in Fig. 3, that displays the outputs from the sensors' network (dots) and from the trained neural network (solid line) used to approximate their dependence on the input parameter. In the subfigures (a)–(h) all the ten outputs of the four sensors are reported. As you can see the perfect match between the sensor outputs and the curve fitting performed by the trained neural network demonstrates the very good performance of the RBF neural network based identification algorithm.

The results of the overall identification process for the system generated by the PDE (45) and the comparison between the distributed and the marginal KLT-based techniques are reported in Fig. 4. Here the inputs to the sensor network (red solid line), the estimated system output (blue dashed line) along with the approximation that would have been achieved by a simple marginal KLT-based (dotted black line) encoding of the system output at the same rate have been displayed. It can be easily seen that a huge improvement of this methodology over the marginal KLT can be achieved for the same rate V.

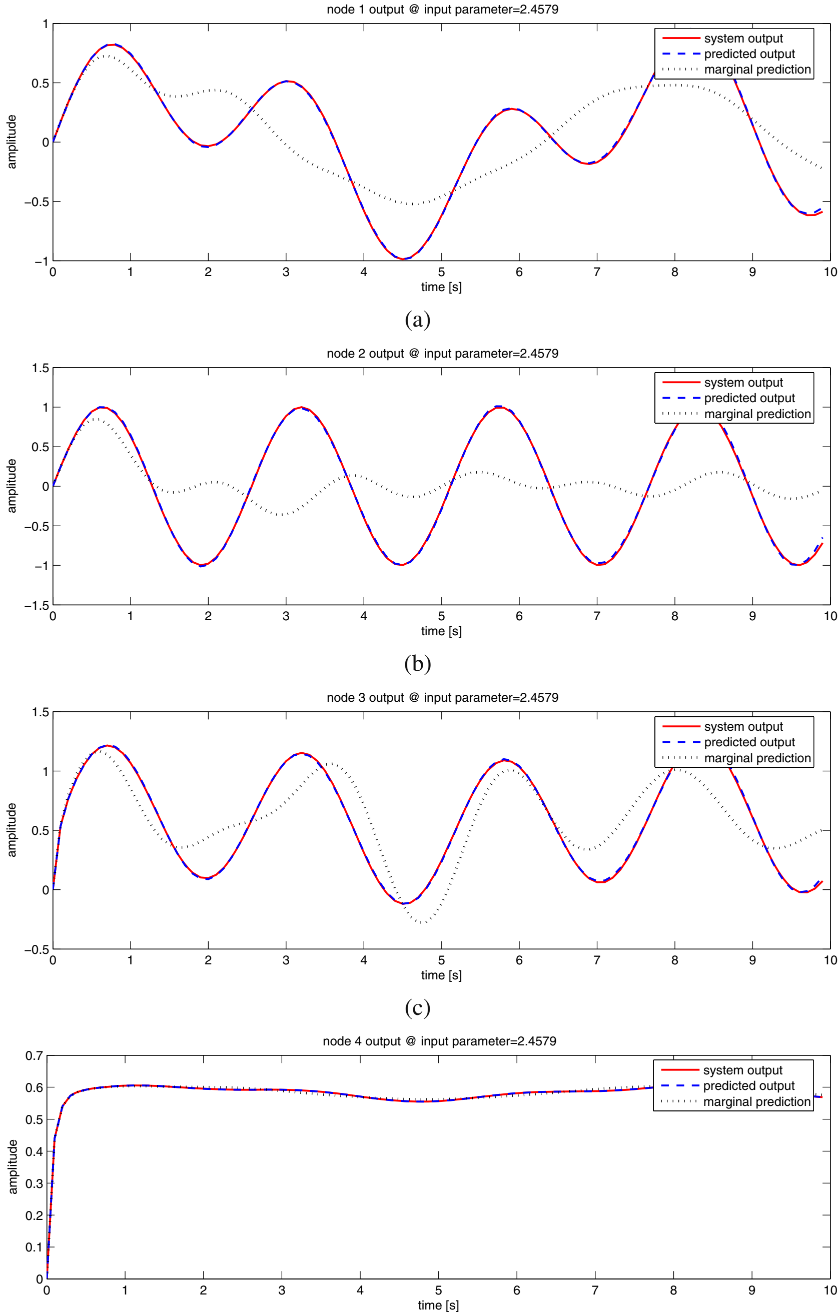
5.2 Second Experiment: Hyperbolic PDE

In this second experiment we tested our algorithm against results obtained with a hyperbolic PDE. A hyperbolic PDE is an equation with spatial operators and second order time derivatives of the form:



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Fig. 3 Outputs from the sensor networks (dots) and from the trained neural network (solid line) used to approximate their dependence on the input parameter. (a) is the only output of sensor 1, (b)–(c) are the two outputs of sensor 2, (d)–(h) the five outputs of sensors 3, and (i)–(j) the two of sensor 4



(a)

(b)

(c)

(d)

Fig. 4 Inputs to the sensor network (red solid line) and estimated system output (blue dashed line). The dotted black line represents the approximation that would have been achieved by a simple marginal KLT-based encoding of the system output at the same rate

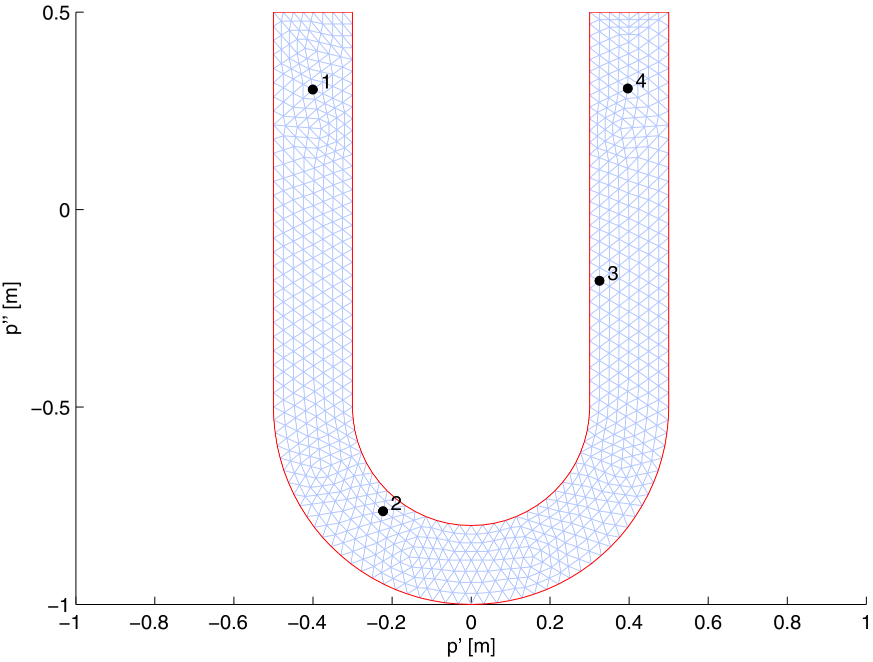


Fig. 5 Domain of the equation used in the second example and mesh used for its solution. Highlighted are the positions in which the sensors were placed

$$d\frac{\partial^2 y}{\partial t^2} - \nabla \cdot (c\nabla y) + ay = f \tag{46}$$

where y is the unknown, d is a complex valued function on a bounded domain $\mathcal{D}$ in the plane, and c, a, f are coefficients that can depend on time t, d can depend on time t as well. The well-known wave equation is a special case of this.

We solved the following PDE,

$$\frac{\partial^2 y}{\partial t^2} - c\,\nabla^2 y = 0 \tag{47}$$

corresponding to (46) with parameters d, c, a, and f, chosen so as to model sound propagation in a U-shaped air duct at a speed $\sqrt{c} = 343$ m/s, on the mesh shown in Fig. 5, where the equation domain $\mathcal{D}$ is also reported. The boundary conditions were set to simulate total reflection at all the surfaces, except at the top left segment that sourced into the domain a sine wave with fixed amplitude and a random frequency $\omega/2\pi$ comprised between 0.5 kHz and 2 kHz. The simulated time-span and the sensor locations were chosen to highlight different phenomena that may occur in wave propagation: sensor 1 is most sensible to the excitation itself and is scarcely influenced by reflections, sensors 2 and 3 are subject to multipath-induced interferences near the bend, and sensor 4 to standing waves.

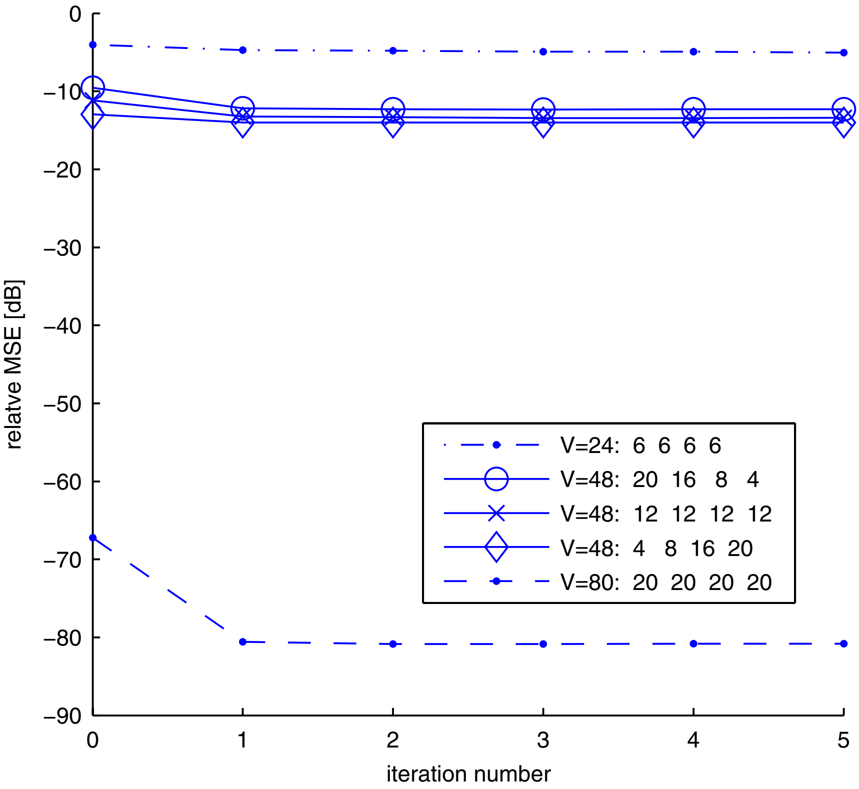
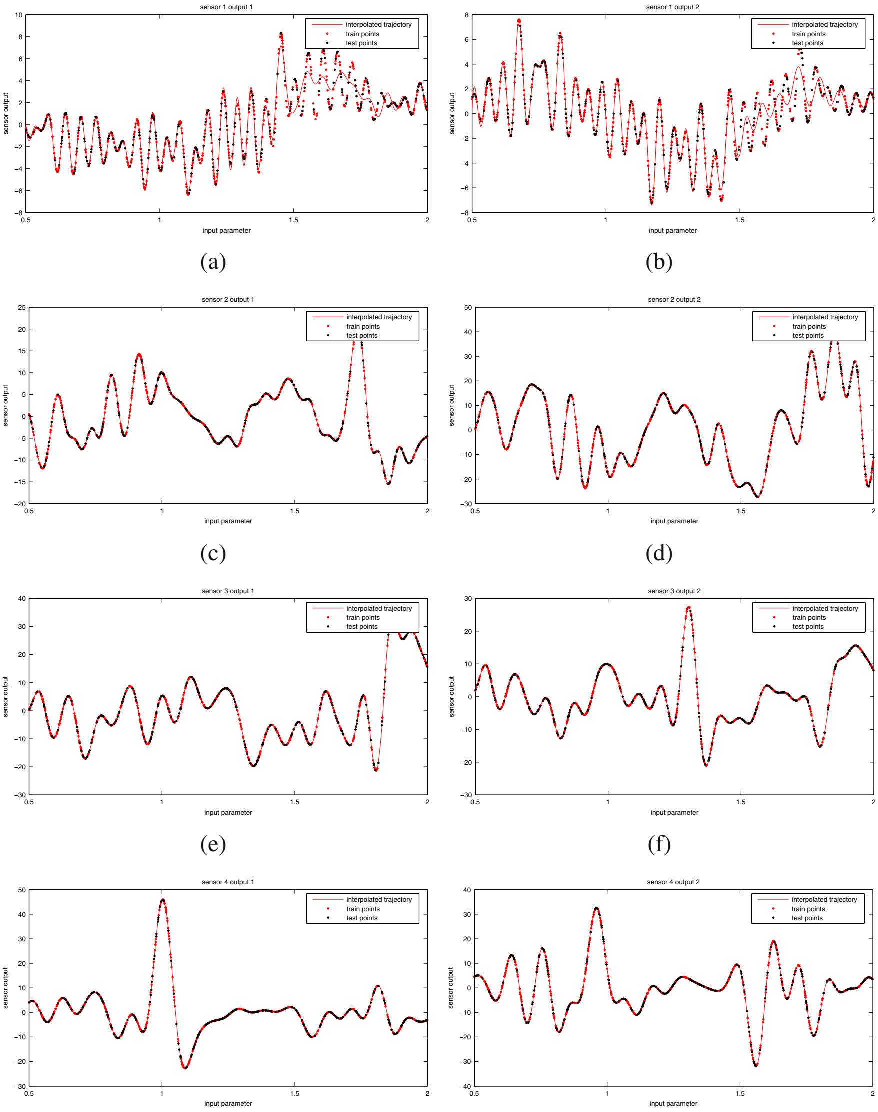


Fig. 6 Relative mean square error of the reconstructed signals plotted versus the number of iterations of the DKLT algorithm, for various rates and rate allocation. The four numbers given in the legend are V_1, V_2, V_3, and V_4, respectively

We conducted various experiments with different rate allocations, with the results being reported in Fig. 6. As it can be easily seen, the rate allocation plays a somewhat important role. For the three cases with a total rate of 48, it is slightly better to allocate more outputs to sensor 4 than to sensor 1, as is reasonable, for it senses the most “complex” signal of the four.

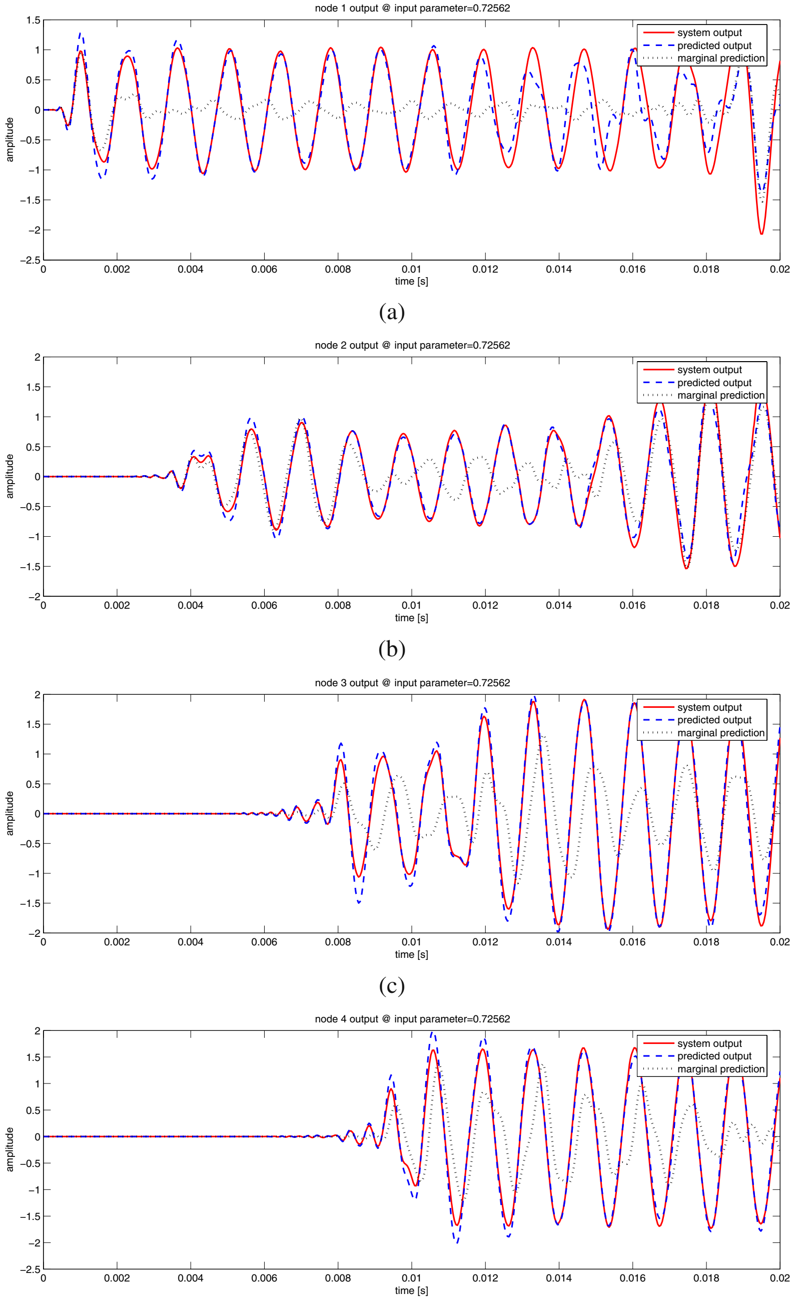
The results for $V_1 = V_2 = V_3 = V_4 = 12$ are shown in Fig. 7, that displays the outputs from the sensors network (dots) and from the trained neural network (solid line) used to approximate their dependence on the input parameter. In the subfigures (a)–(h) the first two outputs of the four sensors are reported. As you can see also in this case the match between the sensor outputs and the curve fitting performed by the trained neural network demonstrates the very good performance of the RBF neural network identification algorithm.

The results of the overall identification process for the system generated by the PDE (47) and the comparison results between our distributed technique and the marginal KLT-based technique are reported in Fig. 8. The inputs to the sensor



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 7 Outputs from the sensor networks (dots) and from the trained neural network (solid line) used to approximate their dependence on the input parameter. (a), (c), (e), and (g) are the most significant outputs of sensors 1–4, respectively, while (b), (d), (f), and (h) are the second significant outputs of the same sensors



(a)

(b)

(c)

(d)

Fig. 8 Inputs to the sensor network (red solid line) and estimated system output (blue dashed line). The dotted black line represents the approximation that would have been achieved by a simple marginal KLT-based encoding of the system output at the same rate

network (red solid line), the estimated system output (blue dashed line), and the approximation that would have been achieved by marginal KLT-based (dotted black line) encoding have been displayed. Also in this case a huge improvement of this methodology over the marginal KLT can be recognized for the same rate V.

6 Conclusions

In this chapter an innovative framework for the collaborative identification of distributed systems has been presented. This approach is based on a centralized intelligent identifier that makes the best identification in a distributed setting on a chosen ensemble of realizations and with no constraints in terms of model kind and/or model order. Methodologically, we defined a stochastic setting where the system to be identified generates nondeterministic signals, i.e., stochastic processes, from given initial conditions and random parameters of input signals. In this way, the set of input-output pairs so obtained shows complex but identifiable geometrical relationships in the output space of the sensor network. As a subsequent step we defined a computational intelligence technique for approximating the previously mentioned mappings that is able to globally identify the distributed system. The global optimization of the identification performance was performed in a collaborative setting, exploiting and developing the cooperation mechanisms that underpin other related methodologies such as the distributed KLT. The effectiveness of the proposed collaborative algorithm has been demonstrated in the identification of two distributed systems whose behavior is described as the solution of a partial differential equation.

References

1. Agre, J., Clare, L.: An integrated architecture for cooperative sensing networks. IEEE Computer 33(5), 106–108 (2000)
2. Baek, S.J., de Veciana, G.: Spatial model for energy burden balancing and data fusion in sensor networks detecting bursty events. IEEE Transactions on Information Theory 53(10), 3615–3628 (2007)
3. Barros, J., Servetto, S.D.: Network information flow with correlated sources. IEEE Transactions on Information Theory 52(1), 155–170 (2006)
4. Belli, M.R., Conti, M., Crippa, P., Turchetti, C.: Artificial neural networks as approximators of stochastic processes. Neural Networks 12(4-5), 647–658 (1999)
5. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
6. Chamberland, J.F., Veeravalli, V.V.: How dense should a sensor network be for detection with correlated observations? IEEE Transactions on Information Theory 52(11), 5099–5106 (2006)
7. Conti, M., Turchetti, C.: Approximation of dynamical systems by continuous-time recurrent approximate identity neural networks. Neural, Parallel & Scientific Computations 2(3), 299–322 (1994)

8. Crippa, P., Turchetti, C., Pirani, M.: Stochastic model of neural computing. In: Proc. Eighth International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2004), Wellington, New Zealand, vol. 2, pp. 683–690 (2004)
9. Cybenko, G.: Approximation by superpositions of sigmoidal function. Math. Control, System, Signal 2, 303–314 (1989)
10. Dana, A.F., Hassibi, B.: On the power efficiency of sensory and ad hoc wireless networks. IEEE Transactions on Information Theory 52(7), 2890–2914 (2006)
11. Dougherty, E.R.: Random Processes for Image and Signal Processing. SPIE—IEEE Series on Imaging Science and Engineering, Bellingham (1998)
12. Dousse, O., Franceschetti, M., Thiran, P.: On the throughput scaling of wireless relay networks. IEEE Transactions on Information Theory 52(6), 2756–2761 (2006)
13. Dukes, P.J., Syrotiuk, V.R., Colbourn, C.J.: Ternary schedules for energy-limited sensor networks. IEEE Transactions on Information Theory 53(8), 2791–2798 (2007)
14. Ferrari, S., Stengel, R.F.: Smooth function approximation using neural networks. IEEE Transactions on Neural Networks 16(1), 24–38 (2005)
15. Franceschetti, M., Meester, R.: Critical node lifetimes in random networks via the chen stein method. IEEE Transactions on Information Theory 52(6), 2831–2837 (2006)
16. Funahashi, K.: On the approximate realization of continuous mappings by neural networks. Neural Networks 2(3), 183–192 (1989)
17. Gastpar, M., Dragotti, P.L., Vetterli, M.: The distributed Karhunen-Love transform. IEEE Transactions on Information Theory 52(12), 5177–5196 (2006)
18. Gastpar, M., Vetterli, M.: On the capacity of large Gaussian relay networks. IEEE Transactions on Information Theory 51(3), 765–779 (2005)
19. Gastpar, M., Vetterli, M., Dragotti, P.L.: Sensing reality and communicating bits: a dangerous liaison. IEEE Signal Process. Mag. 23(4), 70–83 (2006)
20. Georgiadis, L., Tassiulas, L.: Optimal overload response in sensor networks. IEEE Transactions on Information Theory 52(6), 2684–2696 (2006)
21. Girosi, F., Poggio, T.: Networks and the best approximation property. Journal Biological Cybernetics 63(3), 169–176 (1990)
22. Haenggi, M.: On distances in uniformly random networks. IEEE Transactions on Information Theory 51(10), 3584–3586 (2006)
23. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice Hall, Upper Saddle River (1999)
24. Huang, G.B., Chen, L., Siew, C.K.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Transactions on Neural Networks 17(4), 879–892 (2006)
25. Huang, G.B., Saratchandran, P., Sundararajan, N.: A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Transactions on Neural Networks 16(1), 57–67 (2005)
26. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2(3), 395–403 (1989)
27. Kunniyur, S.S., Venkatesh, S.S.: Threshold functions, node isolation, and emergent lacunae in sensor networks. IEEE Transactions on Information Theory 52(12), 5352–5372 (2006)
28. Liu, B., Chen, B.: Channel-optimized quantizers for decentralized detection in sensor networks. IEEE Transactions on Information Theory 52(7), 3349–3358 (2006)
29. Luo, Z.Q.: Universal decentralized estimation in a bandwidth constrained sensor network. IEEE Transactions on Information Theory 51(6), 2209–2210 (2005)
30. Ogunfunmi, T.: Adaptive Nonlinear System Identification — The Volterra and Wiener Model Approaches. Springer, New York (2007)

31. Park, J., Sandberg, I.W.: Universal approximation using radial-basis-function networks. Neural Computation 2(3), 246–257 (1991)
32. Poggio, T., Girosi, F.: Networks for approximation and learning. Proceedings of the IEEE 78(9), 1481–1497 (1990)
33. Scardovi, L., Baglietto, M., Parisini, T.: Active state estimation for nonlinear systems: A neural approximation approach. IEEE Transactions on Neural Networks 18(4), 1172–1184 (2007)
34. Schilling, R.J., Carroll Jr., J.J., Al-Ajlouni, A.F.: Approximation of nonlinear systems with radial basis function neural networks. IEEE Transactions on Neural Networks 12(1), 1–15 (2001)
35. Tay, W.P., Tsitsiklis, J.N., Win, M.Z.: Asymptotic performance of a censoring sensor network. IEEE Transactions on Information Theory 53(11), 4191–4209 (2007)
36. Toumpis, S., Tassiulas, L.: Optimal deployment of large wireless sensor networks. IEEE Transactions on Information Theory 52(7), 2935–2953 (2006)
37. Turchetti, C., Conti, M., Crippa, P., Orcioni, S.: On the approximation of stochastic processes by approximate identity neural networks. IEEE Transactions on Neural Networks 9(6), 1069–1085 (1998)
38. Turchetti, C., Crippa, P., Pirani, M., Biagetti, G.: Representation of nonlinear random transformations by non-Gaussian stochastic neural networks. IEEE Transactions on Neural Networks 19(6), 1033–1060 (2008)
39. Wedge, D., Ingram, D., McLean, D., Mingham, C., Bandar, Z.: On global-local artificial neural networks for function approximation. IEEE Transactions on Neural Networks 17(4), 942–952 (2006)
40. Wu, J.M., Lin, Z.H., Hsu, P.H.: Function approximation using generalized adalines. IEEE Transactions on Neural Networks 17(3), 541–558 (2006)
41. Xiao, J.J., Luo, Z.Q.: Decentralized estimation in an inhomogeneous sensing environment. IEEE Transactions on Information Theory 51(10), 3564–3575 (2005)
42. Xue, F., Kumar, P.R.: On the -coverage and connectivity of large random networks. IEEE Transactions on Information Theory 52(6), 2289–2298 (2006)
43. Yang, Z., Tong, L.: Cooperative sensor networks with misinformed nodes. IEEE Transactions on Information Theory 51(12), 4118–4133 (2005)

Collaboration at the Basis of Sharing Focused Information: The Opportunistic Networks

Bruno Apolloni, Guglielmo Apolloni, Simone Bassis,
Gian Luca Galliani, and Gianpaolo Rossi

Abstract. There is no doubt that the sharing of information lies at the basis of any collaborative framework. While this is the keen contrivance of social computation paradigms such as ant colonies and neural networks, it also represented the Achilles' heel of many parallel computation protocols of the eighties. In addition to computational overhead due to the transfer of the information in these protocols, a modern drawback is constituted by intrusions in the communication channels, e.g. spamming in the e-mails, injection of malicious programming codes, or in general attacks on the data communication. While swarm intelligence and connectionist paradigms overcome these drawbacks with a fault tolerant broadcasting of data – any agent has access massively to any message reaching him – in this chapter we discuss within the paradigm of opportunistic networks an automatically selective communication protocol particularly suited to set up a robust collaboration within a very local community of agents. Like medieval monks who escaped world chaos and violence by taking refuge in small and protected communities, modern people may escape the information avalanche by forming virtual communities that do not in any case relinquish most ITC (Information Technology Community) benefits. A communication middleware to obtain this result is represented by opportunistic networks.

1 Introduction

Beyond its sociological concept, an ideal form of collaboration between computing devices is represented by the paradigm of parallel computing. Having a highly

Bruno Apolloni, Guglielmo Apolloni, Simone Bassis, and Gian Luca Galliani
Department of Computer Science, University of Milan, Via Comelico 39/41,
20135 Milan, Italy
e-mail: apolloni,bassis@dsi.unimi.it,
blapojr@tiscali.it,gian_luca_galliani@tin.it

Gianpaolo Rossi
Department of Information and Technology, University of Milan, Via Comelico 39/41,
20135 Milan, Italy
e-mail: rossi@dico.unimi.it

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 501–524.
springerlink.com

time-consuming computational task, you may think of distributing it on a certain number, say m, of computing devices so that the original running time t becomes t/r with r, also called *throughput*, between 1 and m. A major disappointment of the eighties, when the technology for assembling a huge number of processors was available, came in the realization that, apart from very special tasks, the throughput was never greater than 10 [23, 5]. This is true even with m around 1000, with non rare instances where r is less than 1. The cause of this drawback lies in the overhead introduced in parallel computing by the coordination and synchronization of the single processor tasks. This is achieved mainly through the proper vehicling of the output of a processor to the input of the ones deputed to compute the subsequent tasks.

The *social computation* paradigm [40, 43] overcomes the problem with data transfer protocols that are essentially independent of both tasks and data themselves, and (almost) costless. Whatever its specific goal, a neural network node receives input from the incoming connections as soon as they are available [34], which means either at periodic clock strikes or at an unpredictable (for instance random) time. Analogously, the pheromone deposited by ants is recognized by any other ant passing through the same track [14], and so on. Another feature emerging from these paradigms is the extreme locality of each computation. Like with recent economic theories [44], each processor is an individual taking care of its own computation, possibly the same for all processors, conditioned just by the locally incoming inputs. At a higher level we may modify parameters of these computations, data transmission included, with a uniform procedure, such as *temperature* assessment [31] of the processors or learning algorithms [39], so that a global functionality emerges from the ensemble of the individual processors which complies with our computational goal. Thus, it happens *in spite of* or at least *unbeknown to* individual behavior, simply thanks to smart laws conditioning it. It is not surprising that the favorite way of studying the emerging functionality is a statistical one, and the related distribution laws are either Gaussian or Gibbs distribution laws [20], as descriptors of totally symmetric random phenomena.

Collaboration is something more, even with computing devices. It means that individuals are aware of sharing a common goal with other individuals and jointly use computational and communication tools to achieve it. In this chapter we aim at capturing some mathematical features of collaboration. First of all we realize that collaboration is a process that has a privileged time direction. Moreover, to be effective, any action involves a reaction on the part of collaborating mates, and *vice versa*, so that what emerges from the cooperation, say the cooperation phenotype, is the result of a genotype sequence whose length is in a logarithmic relation with the former. Nested in the social computation framework, we still deal with collaboration in terms of a random phenomenon, and focus on special features of the collaborative actions leading its evolution far from the typical equilibrium distributions to which we are used with physical processes. We are challenged to exploit these peculiarities in a real world scenario that, within the sphere of opportunistic networks [45], gathers both artificial and human collaborating agents in a unified framework. From a true technical perspective, these are telecommunication networks whose exploitation requires the existence of virtual communities. This mix recently won the attention of the scientific community, as a true form of collaborative computational intelligence that is highly promising but still lacks a concrete widespread application.

In very succinct terms, you may think of yourself within a group of agents, each endowed with a short-range radio device able to transmit messages, with a modest energy consumption, preferentially to other agents of the group every time they enter in the operational range of the radio devices. The benefits are clear, covering two aspects: the selective pervasivity of the information broadcasting – i.e. the capability of realizing a capillary information network between agreeing recipients – and the *quasi* gratuity of this operation, as it is automatically managed by low power devices. To get these benefits, however, we must address a certain number of technical problems. You have no guarantee of when and where your message will arrive. Rather, it depends on mates crossing your transmission basin and the paths that they will in turn subsequently follow [10]. Moreover, you cannot transmit the message indefinitely, otherwise you saturate the system capacity and exhaust the power of your device. The strategies you may adopt to solve these problems are principally statistical. Hence they depend on the particular story of the community they refer to – for instance the excursion of a group of tourists, news/info exchanges in a campus, or military men in action – and on the special features you draw from its log – i.e. the statistics on which you base the strategy. As for the latter, with opportunistic networks we face a peculiar framework where the symmetry at the basis of the most widespread probabilistic models is broken by the common interests grouping people within a cluster. In short, in place of a random walk in a public park, people may jointly move toward a cafeteria, though each person reaches it with a randomly perturbed path. This opens scenarios where a basic variable is the intercontact time between transmitting agents, and its distribution law falls in the family of power laws instead of negative exponential laws [21]. From the number of these intercontacts per minute you derive the message speed and the number of hops to reach far away agents in a timely manner. From the reckoning of "infected" agents, i.e. those being reached by a given message, you may decide when to stop transmitting it.

The chapter is organized as follows. We devote the next section both to describing opportunistic networks functionalities and to depicting a possible application scenario which involves the authors research team. Then we formally describe the principled collaboration process and mathematically synthesize its behavior. In this way we introduce new probabilistic scenarios requiring experimental validation. We provide it, at least at a preliminary stage, in Section 3 where we describe an experiment to collect data from the field, providing a statistical analysis. After a brief discussion of the strategies with which to transmit messages with this kind of networks in Section 4, in the last section we draw conclusions and perspectives.

2 The Opportunistic Networks Framework

We may get a quick idea of opportunistic networks by considering a group of people who have short-range radio devices for communicating with one another at a distance somewhat greater than with mere voice range. The device has electronic functions so that the communication is automatic. Hence no one is bored by an intentionally repetitive message sending or storing, and performing, in general, a series of typical database operations. *Vice versa*, people may decide to whom to send the message or conversely from whom to receive a message. Cooperation starts when two people, a sender and a receiver, decide that it is appropriate to forward the message to someone not directly reachable by the sender; rather, its transponding through the receiver device could favor the forwarding. This is the scheme of multi-hop communication, a scheme that is neither robust nor reliable *per se* – i.e. with the task of transmitting a message from *any* agent A to *any* agent B – but may prove extremely compliant with the intents of a virtual community.

According to the literature, we may describe the main features of this framework using the metaphor of *word-of-mouth recommendation* [42] – i.e. a transmission of specific information to selected people, as we may do with private conversations – and the following functionalities of a communication network:

- *device vicinity exploration*. The range limitation of radio-devices requires a *co-location* of transmitting/receiving devices. The movement limitations of their owners – the agents – and their sparseness in a territory identify a speed limitation on the message transmission. Paired with transmission strategies (number of repetitions of the message forwarding and the like) it identifies an effective transmission range [13] of the single agents w.r.t. another one within a group of agents. The interests' coincidence of these agents identifies the range of activity of an opportunistic community;
- *user profile*. A connection between two agents is refused if they do not share a common interest in the message to be transmitted. This implies two levels of message: a *beacon* message[1] to identify the presence of candidate receivers and the preferential transmission band, and the *true* message to be transmitted. We may imagine a graduation of affinity between the mating agents. However, we prefer to discard the role of pure transponding functionality as a service provider [28]. The messages are vehicled exclusively through agents interested in the conveyed information;
- *data dissemination*. In the old children's telegraph game, the kids line up in a row. Then the one in the leftmost position whispers a sentence in the ear of the kid on his right, and so on. The general result is that the last kid on the right hears a sentence that is almost completely different from the original. The cause may be mere background noise or, otherwise, cheating on the part of one or more kids. In the opportunistic network this must not occur. Hence no manipulation is admitted in a message, only its forwarding. The sole corruption may come from the total disappearance of the message, due to the transmission policies [35]. This may be a drawback for some members of the community, but it also represents

[1] In a wireless network, beaconing refers to the continuous transmission of small packets (called beacons) that advertise the presence of a base station (access point). The mobile units sense the beacons and attempt to establish a wireless connection. In opportunistic networks all mobile units may both transmit and sense beacons.

a benefit since it defines a spatial and temporal frontier for the community. It means no spam, information focusing and confidence growth.

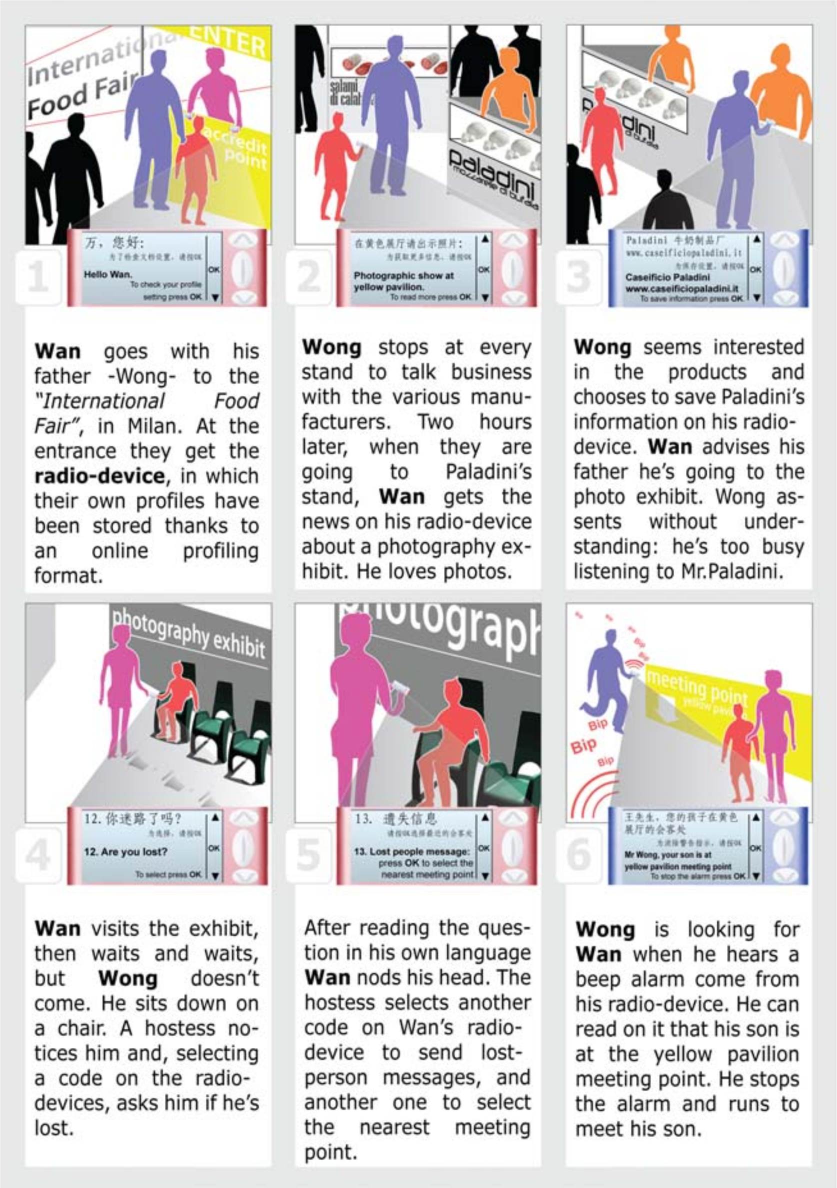
- *privacy/distributedness of the information.* An opportunistic network is a completely distributed system. Each device is the router of the next move with the sole responsibility on the part of the receiver to refuse the communication. To decide a transaction, sender and receiver are anonymous but must declare their interest profile. The degree of identity protection may be more or less high as a function of the cryptographic protocols employed, where a breakeven point must be sought between safety and transmission/computation load. Moreover, we may adopt a reputation system to avoid spammer intrusions [27].

2.1 A Case Study to Lead the Theory

Consider a huge social event, such as a world expo, a championship final, etc. Assume that many tour groups come from around the globe, many of them composed of people familiar with neither local uses nor the native language. There are a lot of reasons for keeping a group of these people in a virtual community held together by an opportunistic network: comfort, security, possibility of fruitful interactions with the environment and so forth. Here below we assume that each member of these groups receives a tourist package upon entering the event site including a radio-device, for short r-d, to state opportunistic connections with his travel companions. Now, from the social design methodology, we borrow a *story board* showing a typical case where the opportunistic community may fulfill a task in an exclusive way that outperforms other solutions. In the forthcoming sections we will give mathematical models and numerical evidences supporting the scenarios.

2.1.1 The Story Board

We follow this story from the different perspectives of a kid and his father from a Far East country, say China, visiting a world expo in Europe as part of a group, and a hostess coming to help them (see the strip in Fig. 1). Both have the above r-d, whereas each of them wants to see and do something different. After his father has been talking about future business deals at the various stands, the son receives on the r-d an announcement about a photography exhibition of interest to him at another pavilion – the yellow one. He tells his father and, assuming he knows, goes to the yellow pavilion. But the father didn’t really get the message, so the face-to-face communication between the two is broken. We may imagine on the one hand the kid waiting for the father in a recognizable puzzled posture (e.g. sitting for a long time looking lost and concerned), on the other the father realizing he has lost the kid when ending one discussion and about to move on to the next scheduled appointment. We may figure many ways of exploiting the r-d facility. One is that a hostess recognizes the kid’s trouble. She doesn’t speak Chinese, but may select a local precoded message on the r-d – say the number 12 : “are you lost?” – which will appear on the LCD both in English and in Chinese, so that it can be understood by the kid.



Wan goes with his father -Wong- to the *"International Food Fair"*, in Milan. At the entrance they get the **radio-device**, in which their own profiles have been stored thanks to an online profiling format.

Wong stops at every stand to talk business with the various manufacturers. Two hours later, when they are going to Paladini's stand, **Wan** gets the news on his radio-device about a photography exhibit. He loves photos.

Wong seems interested in the products and chooses to save Paladini's information on his radio-device. **Wan** advises his father he's going to the photo exhibit. Wong assents without understanding: he's too busy listening to Mr.Paladini.

Wan visits the exhibit, then waits and waits, but **Wong** doesn't come. He sits down on a chair. A hostess notices him and, selecting a code on the radio-devices, asks him if he's lost.

After reading the question in his own language **Wan** nods his head. The hostess selects another code on Wan's radio-device to send lost-person messages, and another one to select the nearest meeting point.

Wong is looking for **Wan** when he hears a beep alarm come from his radio-device. He can read on it that his son is at the yellow pavilion meeting point. He stops the alarm and runs to meet his son.

Fig. 1 Strip depicting lead story from different perspectives

Then the same device may be used to forward both an emergency message to the leader of the group (assumed to be nearby) and to the expo security staff, as well as a specific message to the relatives of the kid who will recognize the message ID. Both messages should ensure a happy ending.

This simple case highlights some strategic features expected from the r-d:

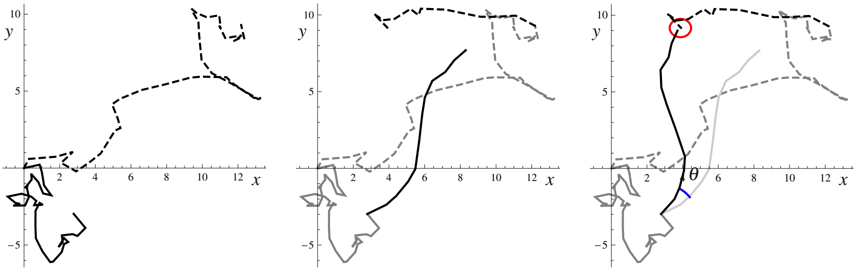
1. Beside the radio waves, the local communication channels of the device should be both audio, for emergency signal, and visual, committed to a small LCD screen. Three buttons: up, down and ok, will manage the communications.
2. The memory of the device should store an ID of its user, for instance made up of: language, age, sex and preference re the message content he likes to receive. The storing may be done either in remote, via the Internet, or directly upon device delivery.
3. In addition, the user may decide to store special messages he receives that may represent a record of his visits and hints about future tourist proposals.
4. The opportunistic functionality of the device concerns the circulation of service and emergence messages. They are improved by some local functionalities, consisting of translation of pre-coded inquiries that may be either sent in remote or formulated locally (such as "are you lost?").

2.2 An Elementary Mathematical Model

The intercontact time is the key ingredient of this networking mode which has quite recently gained the attention of the researchers. Studying it calls for an extensive comprehension of the users' mobility, a task far from fulfillment. Since it involves random phenomena, we find in the literature both statistical analyses of the message transmission traces, and probabilistic models of the message piping dynamics. Let us visualize the mathematical terms of our approach through the example of dodgem cars. Although the primary goal of each player is to hit the car of another player, this example correctly highlights the intentionality at the basis of the collaborative task under consideration. Assume you are playing with dodgem cars at an amusement park. You drive around until, from time to time, you decide to bang into a given car which is unaware of your intent. For the sake of simplicity we may assume the trajectory of each car to be a plane Brownian motion before the chase is triggered. Thus, with the reference frame in Fig. 2(a), indexing with $i = 1,2$ the cars whose stories we are following, we have:

$$X_i(t) \sim \mathscr{N}_{0,\sqrt{t}} \qquad Y_i(t) \sim \mathscr{N}_{0,\sqrt{t}} \tag{1}$$

where $\mathscr{N}_{\mu,\sigma}$ is a Gaussian variable of mean μ and standard deviation σ. Then you, sitting in the first car, decide at time τ to reach and crash into the second car. The questioned variable records the instant $T > \tau$ when you succeed. In the case study where cars are points in the plane, in order to identify this instant we must specify: i) an operational definition of the cars' clash since the probability of exact matching is 0, and ii) the symmetry break introduced by the chase intention. The chase effectiveness depends on the capability of orienting your motion in the direction of the target, which corresponds to converting a part of the motion along the cars' connecting line from symmetric to oriented moves. Mathematically, orientation



(a) (b) (c)

Fig. 2 Joint traces of two cars (plain and dashed curves respectively) when: (a) both move according to a Brownian motion behavior; (b) the former moves only in one quadrant (absolute value of the Brownian motion components) from a trigger time on; and (c) an oracle rotates this trajectory toward the other car with some approximation (quantified by the ray of a proximity circle)

corresponds to taking the absolute value of the elementary steps in that direction, so as to work with Chi distributed addends in place of Gaussian ones (see Fig. 2(b)).

In order to overcome analytical complications we propose this simple scheme. As the difference between two Gaussian variables is a Gaussian variable too, we may use (1) also to describe the components of the distance Δ between the two cars before τ. We just need to multiply them by $\sqrt{2}$ so as $X_\Delta(t) \sim \mathcal{N}_{0,\sqrt{2t}}$ and similarly for $Y_\Delta(t)$. Moreover, if we move to polar coordinates (r,θ) with $x = r\cos\theta, y = r\sin\theta$, the density function f_Δ of Δ becomes

$$f_\Delta(r,\theta) = \frac{1}{4\pi t} r \mathrm{e}^{-\frac{r^2}{4t}} \tag{2}$$

which looks for the joint density function of (R,Θ), with R a Chi variable with 2 degrees of freedom scaled by a factor $\sqrt{2t}$ and Θ a variable uniformly distributed in $[0,2\pi)$. Our assumption about the pursuit is that, with reference to the distances D_1 and D_2 of the two cars from the position of the first one at time τ, you are able to maneuver Θ_1 from τ on, so that when $D_1 = D_2$ also $\Theta_1 = \Theta_2$ (see Fig. 2(c)). As mentioned before, *per se* the probability of a match between two points representing the cars is null. Thus your task is unrealistic. However, intentionality recovers feasibility thanks to the fact that in practice it is enough that the angles are sufficiently close to entangle the two cars. The actual correspondence with the pursuit dynamics is facilitated by some free coefficients which will be embedded in the model.

With this assumption we are interested in the time t when $D_1 = D_2$. Given the continuity of the latter we may measure only a probability density with t. In other words, at any change of the sign in the difference $D_1 - D_2$ with the running of the two cars, there will correspond a matching time as a specification of a continuous variable T. Since both D_1 and D_2 scale with the square root of time, expressing their dependence on the *trigger time* τ and the *pursuit time* t, we have

$$D_1(t) = \sqrt{t}\chi_{2_1}; \qquad D_2(t) = \sqrt{2\tau+t}\chi_{2_2} \tag{3}$$

where χ_2 is a random variable with density function $f_{\chi_2}(z) = ze^{-\frac{z^2}{2}}$. Thus, after equating $D_1(t)$ with $D_2(t)$ we obtain

$$1 = \frac{D_1(t)}{D_2(t)} = \frac{\chi_{2_2}}{\chi_{2_1}} \frac{\sqrt{2\tau+t}}{\sqrt{t}} \tag{4}$$

under the condition $\chi_{2_2} \geq \chi_{2_1}$. Denoting with T the random variable with specifications t and $\mathscr{T}$ with specifications τ, this equation finds a stochastic solution in the random variable

$$V = \frac{T}{\mathscr{T}} = 2\left(\frac{\chi^2_{2_1}}{\chi^2_{2_2}} - 1\right)^{-1} \tag{5}$$

It follows the same distribution law of the ratio between two unconstrained Chi square variables, i.e. an F variable with parameter $(2,2)$ [19], whose cumulative distribution function (CDF) reads

$$F_V(v) = 1 - \frac{1}{1+v} I_{[0,\infty)}(v) = \frac{v}{1+v} I_{[0,\infty)}(v) \tag{6}$$

where $I_{[a,b]}(x)$ is the indicator function of x w.r.t. the interval $[a,b]$, thus being 1 for $a \leq x \leq b$, 0 otherwise.

2.3 A Very General Way of Maintaining Memory in a Time Process

Let us make some general considerations about processes with memory. To start from the very beginning, for any *ordered* variable T, such that only events on their sorted values are of interest to us, the following *master equation* holds

$$\mathrm{P}(T > t | T > k) = \mathrm{P}(T > q | T > k)\mathrm{P}(T > t | T > q) \quad \forall k \leq q \leq t \tag{7}$$

It comes simply from the fact that in the expression of the conditional probability

$$\mathrm{P}(T > t | T > k) = \frac{\mathrm{P}(T > t)}{\mathrm{P}(T > k)} = \frac{g(t)}{g(k)} \tag{8}$$

we may separate the conditioned variables from the conditioning ones. While (7) denotes the time splitting in the fashion of the Chapmann–Kolmogorov theorem [37] as a general property of any sequence of data, equation (8) highlights that events $(T > t)$ and $(T > k)$ are by definition never independent. What is generally the target of the memory divide in random processes is the time $t - k$ elapsing between two events. In this perspective, the template of the memoryless phenomena descriptor is the Poisson process, whose basic property is $\mathrm{P}(T > t) = \mathrm{P}(T > q)\mathrm{P}(T > t - q)$, if $t > q$. It says that if a random event (for instance a hard disk failure) did not

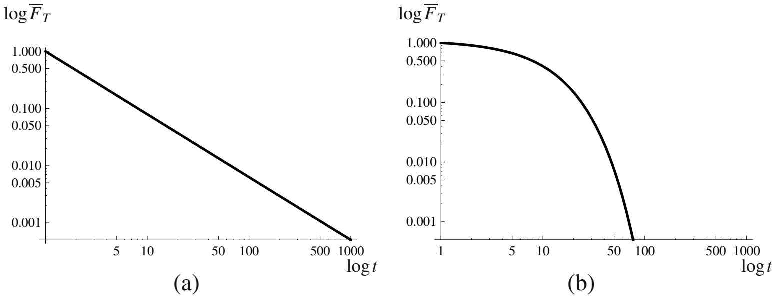


Fig. 3 CCDF LogLogPlot when T follows: (a) a Pareto law with $\alpha = 1.1$ and $k = 1$; (b) a negative exponential law with $\lambda = 0.1$. Parameters were chosen to have the same mean 11

occur before time q and you ask what will happen within time t, you must forget this former situation (it means that the disk did not become either more robust or weaker), since your true question concerns whether or not the event will occur at a time $t-q$. Hence your true variable is $\tau = T - q$, and the above property is satisfied by the negative exponential distribution law with $\mathrm{P}(T > t) = \mathrm{e}^{-\lambda t}$, for constant λ[2], since with this law (7) reads

$$\mathrm{e}^{-\lambda(t-k)} = \mathrm{e}^{-\lambda(q-k)}\mathrm{e}^{-\lambda(t-q)} \tag{9}$$

and the property that $\frac{g(t)}{g(k)}$ in (8) equals $g(t-k)$ is owned only by the exponential function.

In contrast, you introduce a memory of the past (q-long) if you cannot separate $T - q$ from q. In this paper we consider very simple cases where this occurs because the time dependence is of the form $\tau = (T/q)^{\beta}$. The simplest solution of (7) is represented by $\mathrm{P}(T > t | T > k) = (t/k)^{-\alpha}$ so that the master equation reads

$$(t/k)^{-\alpha} = (t/q)^{-\alpha}(q/k)^{-\alpha} \tag{10}$$

Note that this distribution, commonly called the Pareto distribution, is defined only for $t \geq k$, with $k > 0$ denoting the true time origin, where α identifies the distribution with the scale of its logarithm. The main difference w.r.t. the negative exponential distribution is highlighted by the LogLogPlots of the complementary cumulative distribution function (CCDF) $\overline{F}_T$ in Fig. 3: a line segment with a Pareto curve (see picture (a)) in contrast to a more than linearly decreasing curve with the exponential distribution (picture (b)).

The difference between the graphs in Fig. 3 shows that, for a same mean value of the variable, we may expect this occurrence in a more delayed time if we maintain memory of it as a target to be achieved rather than if we rely on chance.

We recover the distribution of V coming from the dodgem car model by extending (6) as follows

[2] Variants with $\lambda = 1/\beta(t)$ allow simple adaptation of the law to more complex phenomena when $\beta(t)$ is not far from being a constant.

$$F_V(v) = 1 - \frac{b}{b + (v/c+1)^a} \tag{11}$$

which we call *shifted* Pareto, with b and c playing the role of both scale and shift parameters. The latter stands for a key feature of a memory dependent model; the former of a memoryless framework. The exponent a plays a role similar to α's. With this formula we aim to approximately capture many variants of v, as for both modeling variations and model adaptation to real data. For instance we almost recover

$$F_V(v) = \frac{2v^{\frac{2}{\alpha}}}{v^{\frac{2}{\alpha}} + (v+2)^{\frac{2}{\alpha}}} I_{[0,\infty)}(v) \tag{12}$$

obtained by extending the dependence on t from square root to a generic power $1/\alpha$ in model (1). Though somewhat structurally different from (12), (11) almost coincides with (12). In particular, with $a = b = c = 1$ and $v = v - 1$ it takes the form (6) into which (12) translates when $\alpha = 2$. Actually, we may get satisfactory approximations in a relatively wide range of parameters. Moreover V in (11) ranges from $-c$ to $+\infty$. Hence, when we refer to a variable in $[0, +\infty)$, we use the truncated version of (11) that is given by

$$F_V(v) = 1 - \frac{(b+c)}{b + (v/c+1)^a} \tag{13}$$

To obtain the pursuit times we need to multiply v by the trigger time; we must also add the latter to the product in order to obtain the contact times. In the next section we will see that both contact times and intercontact times remain approximately in the same family, provided we have a suitable distribution law of the trigger times. We will also study some manageable deviations from this model.

2.4 The Timing of the Intentional Process

From our model we are left with a power law describing the ratio between pursuit and trigger times. Since $t = v\tau$, to complete the description of contact times we need a model for the trigger time too. Let $f_{\mathscr{T}}$ be its probability density function (PDF), defined in a range $(\tau_{\text{inf}}, \tau_{\text{sup}})$. Since $t + \tau = (v+1)\tau$, we obtain F_W with $W = T + \mathscr{T}$ by computing

$$F_W(w) = \int_{\tau_{\text{inf}}}^{\max\{w, \tau_{\text{sup}}\}} F_V(w/\tau - 1) f_{\mathscr{T}}(\tau) \mathrm{d}\tau \tag{14}$$

With many families of $\mathscr{T}$ PDFs the shape of V CDF transfers to W, as shown in Figs. 4(a) to (c). The difference of the trigger time distributions slightly affects the length of the plateau, the slope of the linear trait and the smoothness of their unions as well, all elements that make great differences in the linear scales.

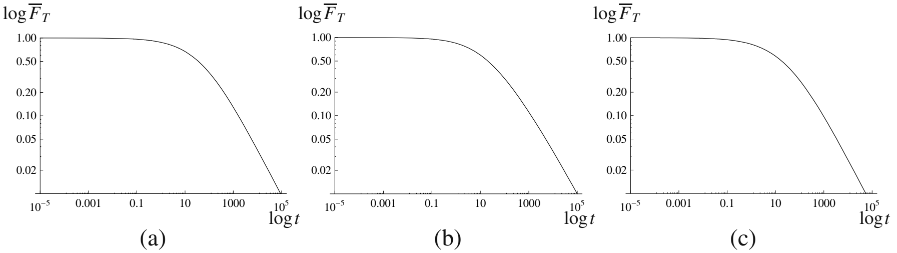


Fig. 4 CCDF LogLogPlot of contact times with a trigger time varying according to distribution law: (a) uniform ; (b) Pareto; and (c) negative exponential

For instance, for a Pareto distributed trigger time:

$$F_{\mathscr{T}}(\tau) = 1 - \tau^{-\lambda} \tag{15}$$

with $c = 1$ we have the expression

$$F_W(w) = 1 - {}_2\mathscr{F}_1\left(1, \frac{\lambda}{a}; \frac{a+\lambda}{a}; -\frac{w^a}{b}\right), \tag{16}$$

where ${}_2\mathscr{F}_1$ is the Hypergeometric2F1 function defined as ${}_2\mathscr{F}_1(a,b;c;z) = \sum_{k=0}^{\infty} \frac{(a)_k(b)_k}{(c)_k}\frac{z^k}{k!}$, with $(a)_k = a(a+1)(a+2)\cdots(a+k-1)$ being the rising factorial [1].

But the time until trigger is not an idle time. Some encounters may occur even during the initial wandering. We prefer to study this T contribution empirically. For instance, In Fig. 5 we see the empirical complementary cumulative distribution function (ECCDF) LogLogPlot of a T sample obtained by tossing a trigger time uniform in $[1, 1000]$. To these points we added further samples coming from an exponential distribution with tentatively $\lambda = \alpha$ exactly truncated to the maximum trigger time. These points take into account the casual crossing of a device in the range of another one during its non intentional motion. The ratio between points sampled before and after trigger times is a way of fixing the true λ. We see a plateau analogous to the one in Fig. 4, having the abscissa of its right end around 100, followed by a linear slope. This distribution may be recovered through a Pareto-like distribution of form (13) apart from the small hump over the sloping trait, which we realize is still an effect of the exponentially drawn points and will represent a peculiarity of the experimental curves to be seen in the next section. Indeed, the shape of (13) presents the plateau shown in these diagrams and a linear trait constituting the second main feature of them. In particular, the parameter $b^{1/a}$ (where $a = 1.1$) is strictly connected to the average trigger time, denoting the divide between non intentional and intentional behavior, a is proportional to the rate of contact when agents move according to some purpose, and c is a fine-tuning factor.

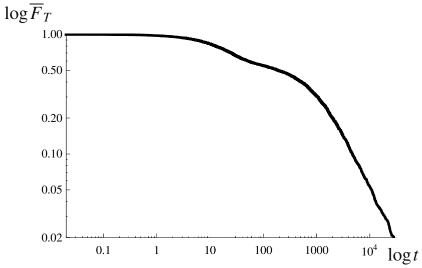


Fig. 5 Recovering the intercontact ECCDF shape through our mobility model

3 Validating the Model

As we may argue from the previous sections, opportunistic rendezvous involve complex processes that depend highly on the operational context. However, analytical forms like (13) seem to denote that there is a basic course which characterizes the intercontact time T. We may also guess that this behavior biases the other variables that boil up around this particular communication network. That is why a certain number of theories have been constructed around T, and companion experimental campaigns have been carried out to validate the models.

3.1 Drawing Data and Models from the Literature

We can presently find in the literature a few real user-traces databases which we generally consider limited, specific to particular environments and difficult to manage. However, a common feature emerging from these traces is that the distributions underlying mobility phenomena are heavy-tailed distributions having the Pareto law as a common template [7, 25, 33, 30, 10]. Actually, the CCDF synthesizing these tracks shows two different traits that appear linear (see Fig. 6(a)), thus suggesting descriptions in terms of: a *double Pareto* curve (a lower power curve followed by a greater power one), or, alternatively, a temporal sequencing of a Pareto trait prosecuting with an exponential distribution that quickly nears 0 [29] (see Figs. 6(b) and (c), respectively); or, in contrast, the sequencing of an exponential trait and a Pareto trait as we suggested in the previous sections.

This behavior has no well-assessed theoretical model in the literature. So, synthetic models normally produce quite different time distributions which find justification more in human mobility abstract (possibly simplifying) hypotheses than in experimental feedbacks. Thus, at a very elementary level, we find a Random Walk Mobility Model [8] described in terms of a Brownian process [17], where each user randomly chooses a direction and a speed at each time-step. The interaction between users is enhanced in two directions: i) the dynamics of the single agent, and ii) the correlation between their moves. As for the former, in the Random

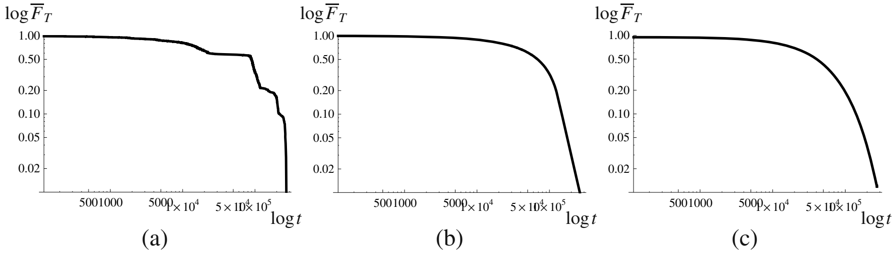


Fig. 6 CCDF LogLogPlot of a T randomly varying according to: (a) the Cambridge/Haggle Crawdad dataset; and (b) a double Pareto law; and (c) a power law followed by an exponential one

WayPoint Mobility Model [48, 26] each user randomly selects a target point in the region where users move and goes toward it (the *pursuit* phase) with a speed uniformly chosen in a fixed interval. Once he arrives he remains there for a fixed time and then starts moving again with parameters drawn independently from the previous ones. The model is made more sophisticated in various ways. For instance, according to the Gauss-Markov model, the direction and speed at time t depend on their values at time $t-1$ [36]; otherwise the path toward the goal region takes into account the obstacles represented by buildings [12] etc. As for the second direction, correlations between users are considered in the Group Mobility Model and its variants [9].

3.2 A Homemade Validation

Aiming for a more direct feedback on our model, we decided to collect and analyze human mobility traces in a campus area. We achieved this by developing portable radio devices, denoted as Pocket Traces Recorders, and by deploying different test beds involving faculty members and students.

3.2.1 Requirements

The design of the Pocket Trace Recorder, or PTR, has both functional and architectural requirements. The former are related to trace collection, recording and transferring to a server station for off-line analysis. The primary focus of the PTRs design is the collection of data that describe the contacts among encountering devices. The distribution of the intercontact times between mobile devices is the key ingredient to estimate the delays in the system and to generate a mobility model seamlessly reproducing real human mobility. On the other hand, the amount of contacts and how long they last, for both individual and group interactions, provide significant information about the network capacity and the people's way of moving according to some spatial, social or functional law [36].

The main PTR architectural requirement is to enable experiments to last 3-4 weeks without needing human intervention for any battery changes. This energy consumption constraint heavily influenced the design of the PTR and forced a departure from other experiences that placed more computation power onboard and adopted more energy consuming radio technologies [18]. PTRs can operate with a configurable frequency with which beacons are transmitted. The corresponding time periods range from 1 second to a few minutes, according to the mobility environment we wish to observe. Once defined at configuration time, however, the beacon time may autonomously range within a small time interval to grant access to the channel in crowded locations. After sending its beacon, a PTR enters a sleep mode and wakes up whenever it receives a beacon from the neighborhood.

Whenever a beacon is received from a given encounter, let us say PTRe, the device creates a new entry in the local *contact-log* if no entry is active for PTRe. The beacon is discarded otherwise, while the entry is maintained and contains the following items: i) local ID and ID of the encounter PTR; ii) the time stamp of the first contact; and iii) the time stamp of the contact closing event. An entry in the contact-log is *closed* when the beaconing from the encounter device has been missing for more than t seconds, with $t = 60$ seconds in our experiments. The local memory size should be dimensioned to store the contacts of experiments lasting up to 3-4 weeks. Our test beds have generated on average 2000 contacts per device, with beaconing time set to 1.5 seconds. The drawback of using a non-standard radio access technology, as opposed to a general standard such as Bluetooth, is to be unable to record the contacts with other mobile devices in the area. Finally, no specific bandwidth and processing requirements have been envisaged for PTRs.

3.3 The Architecture

The Pocket Trace Recorder is a portable radio device with the overall architecture described in Fig. 7. It uses the Cypress CY8C29566 microcontroller and the radio module AUREL, model RTX-RTLP. The radio range has been limited to 10 meters in order to maintain a sparse PTR distribution even in an office area and to limit power consumption. This combination allows a very low power consumption that lets the experiments last for the required time with common batteries NiMh, AA $1.2V$. Each PTR has a 1 MB flash memory where more than $50,000$ contacts can be stored.

The PTR firmware implements the first two layers of ISO-OSI model [32]: Manchester coding is used at the physical layer, while a CSMA non-persistent MAC protocol that regulates access to the $2400b/s$ channel characterizes the data-link layer. Within the latter layer, beacons are the only frames that a device has to exchange with its neighbors. The beacon payload is composed of: the PTR identifier; a set of bits representing the internal state (and which are used for diagnosis purposes); and the current time. The local time is set at the configuration time. The total

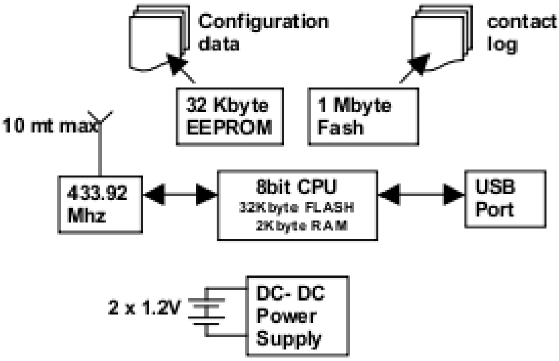


Fig. 7 PTR architecture

clock drift in 3-week experiments has been evaluated in 10-12 seconds and, so far, we have not executed any run-time clock synchronization. After sending its beacon, a PTR enters a sleep mode and wakes up whenever it receives a beacon from the neighborhood.

Each PTR uses an USB interface to communicate with the Pocket Viewer, the Desktop application software, which has been used to configure the devices, collect the recorded data at the end of the experiment and support data analysis and device monitoring.

3.4 Preliminary Matches

We arranged the collected data so as to analyze intercontact and intracontact times. To remove artifacts, we remodulated the recorded times by eliminating idle periods represented by the time intervals where people are expected to be far from the campus. Namely we contracted to 0 the time intervals between 7 p.m and 8 a.m. of workdays and the entire weekends. We also clamped to 0 the last 60 seconds of contacts that we essentially assume to be artificially generated by the above beaconing control rule. After this preprocessing, we have for each PTR a log of its inter- and intracontact times with any other of the PTRs enrolled in the experiment. Thus we may focus on three kinds of statistics, respectively concerning: single pairs, one PTR versus the remaining ones, and each PTR versus others. As the second kind concerns the union of the files singularly collected for the first, and analogously with the third w.r.t. the second kind, we are essentially considering the same distribution laws but with different parameters. The basic distributions – negative exponential and Pareto – are indeed reproducible distributions [47]. Thus, the minimum among a sample of sampled values has the same CCDF as the original values in both cases, but with possibly changed parameters.

From previous sections we are left with: i) a constructive model to produce empirical CCDFs, and ii) an approximate analytical counterpart having the benefit of

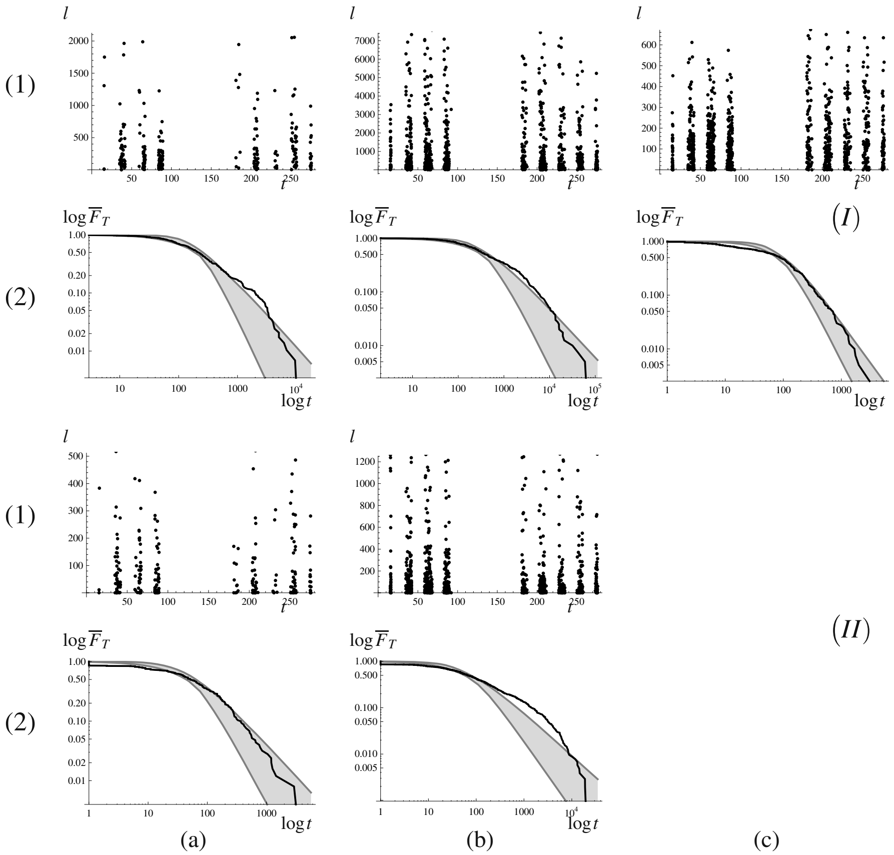


Fig. 8 Overview of the PTR connections' timing. Part I: Intercontact times; Part II: intracontact times. On each part, line (1): episode lengths l vs. their starting time t (in hours); line (2): episode lengths ECCDF with time expressed in seconds; column (a): encounters between a specific PTR pair; column (b): encounters of a specific PTR with all remaining ones; column (c): only for intercontacts, net intercontact times independently of the mate ID

being characterized by a reduced number of parameters (see (13)) that are relatively easy to infer. Fig. 8 reports an excerpt of the experimental distributions. Namely we see: 1) the log of intercontact and intracontact lengths with the daily times, and 2) their synthesis in terms of ECCDF. For each item we have an exemplar related to the above three layers: single pair, one versus others and each versus every other. The type (a) diagrams highlight a common path that the PTR encounters during the course of a day, whereas a more detailed analysis could reveal biases with the weekdays and preferences of the single PTR owners. All these biases seem however to be well adsorbed by the Pareto-like distribution as shown in Fig. 8. Here the experimental graphs are satisfactorily recovered with curves as in (13) through confidence regions (whose computation together with a general inference procedure are remanded to the next section), apart from a slight hump overposition that we

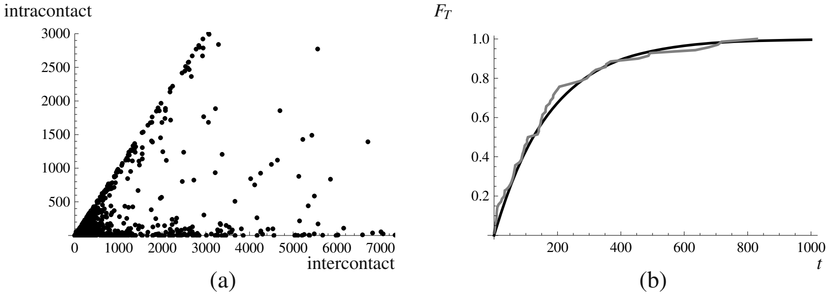


Fig. 9 The non intentional features of the encounters. (a) Scatterplot of intercontact-intracontact times drawn from same records as in column (b) of Fig. 8. (b) Exponential CDF of the first intercontact times when the contacts have almost zero duration

already observed on the sloping trait of the picture in Fig. 5 and interpret through the simple constructive model in the previous section.

We obtained feedback from the experimental data on the matter presented through the pictures in Fig. 9. From part (a) of this figure – representing the scatterplot of intercontact-intracontact times – we see that most of the intracontact times lasting less than 2 seconds are linked to small intercontact times, actually those located within the plateau. Whereas the high correlation between the two times (around 0.27) depends on the PTR blindness to other contacts once one contact has been stated, the distribution of these times within the plateau when the intracontact time is less that 2 seconds means the encounters are non intentional. Indeed, as shown in Fig. 9(b) the distribution is exponential, figuring only crosswalks that are random and have an almost null duration.

3.5 *The Statistical Versant*

Inferring either a Pareto or a Pareto-like distribution is not a standard task *per se*. In addition, we must consider that empirical data are affected by many artifacts, linked for instance to seasonal phenomena such as user habits during a particular week and/or on a particular day of the week, special tasks shared exclusively by some pairs of users, etc. Thus, rather than expecting a perfect fit, we look for tight regions such as confidence regions [3] in which the experimental curves can be expected to lie with a high probability. The identification of these regions is a favorite task of the Algorithmic Inference paradigm [4, 2]. We infer them via a bootstrap procedure [16] that we develop through the following steps.

1. *Sampling mechanism.* It consists of a pair $\langle \Psi, g_\theta \rangle$, where the *seed* Ψ is a random variable without unknown parameters, while the *explaining function* g_θ is a function mapping from samples of Ψ to samples of the random variable X we are interested in. Thanks to the probability integral transformation theorem [38] we have that, by using the uniform variable U in $[0,1]$ as a seed Ψ, the explaining function for T distributed according to (13) is:

$$t = c\left(\frac{1+bu}{1-u}\right)^{\frac{1}{a}} - c \tag{17}$$

2. *Master equations.* The actual connection between the model and the observed data is tossed in terms of a set of relations between statistics on the data and unknown parameters that come as a corollary of the sampling mechanism. With these relations we may inspect the values of the parameters that could have generated a sample with the observed statistic from a particular setting of the seeds. Hence, if we draw seeds according to their known distribution – uniform in our case – we get a sample of compatible parameters in response [4]. In order to ensure this sample clean properties, it is enough to involve sufficient statistics w.r.t. the parameters [41] in the master equations. Unluckily, because of the shift terms, the parameters are so embedded in the density function of T that we cannot identify such statistics for them. Rather we focus on the statistics $t_{\text{med}} = t_{(\lfloor m/2+1 \rfloor)}$, $s_1 = \sum_{i=1}^{m} \left| t_{(i)}/t_{\text{med}} - 1 \right|^r$ and $s_2 = \sum_{i=\lfloor m/2+1 \rfloor}^{m} \log\left(t_{(i)}/t_{\text{med}}\right)$, where $t_{(i)}$ denotes the i-th item within the sample sorted in ascending order, and propose the following master equations having analogous notation for $u_{(i)}$ and u_{med}

$$t_{\text{med}} = c\left(\frac{1+bu_{\text{med}}}{1-u_{\text{med}}}\right)^{\frac{1}{a}} - c \tag{18}$$

$$s_1 = \sum_{i=1}^{m} \left| \left(\frac{\left(\frac{1+bu_{(i)}}{1-u_{(i)}}\right)}{\left(\frac{1+bu_{\text{med}}}{1-u_{\text{med}}}\right)} \right)^{\frac{1}{a}} - 1 \right|^r \tag{19}$$

$$s_2 = \sum_{i=\lfloor \frac{m}{2}+1 \rfloor}^{m} \log\left(\frac{1+bu_{(i)}}{1-u_{(i)}}\right) - \log\left(\frac{1+bu_{\text{med}}}{1-u_{\text{med}}}\right) + \left\lfloor \frac{m}{2}+1 \right\rfloor \log c \tag{20}$$

We cannot state a general course of these statistics with the parameters. However, in the region where we expect to find a solution this course is monotone in a and b, with a tilting attitude of c. In greater detail, the true ratio term in s_1 and s_2 should be $\left(\left(\frac{1+bu_{(i)}}{1-u_{(i)}}\right)^{\frac{1}{a}} - 1\right) \Big/ \left(\left(\frac{1+bu_{\text{med}}}{1-u_{\text{med}}}\right)^{\frac{1}{a}} - 1\right)$, which we approximately simplify as in (19) to make the solution easier. Moreover, the last term in (20) does not derive from the explaining function of T. Rather, it is added to introduce a shaking in the numeric search of a solution, in the assumption that c is close to 1, hence its logarithm to 0, while shifts from this value raise t_{med} and decrease s_2. To achieve the first condition together with a suitable range of values, we normalize the t_is with a wise shift and rescaling. We also modulate r, ruling a balance between second and third statistics, in order to avoid disastrous paths followed by the search algorithm leading to diverge. This denotes the fact that at the moment, due to the difficulties of the involved numerical optimization tasks, the estimation procedure is not completely automatic.

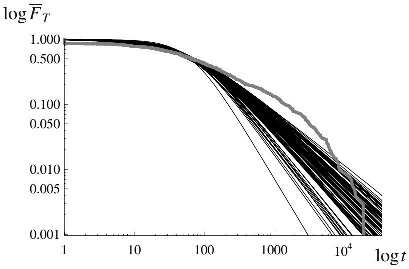


Fig. 10 Population of fitting curves

Embedding the found parameters in (13) we obtain a population of random curves to contrast with the empirical one as in Fig. 10. From the empirical population a confidence region may be derived, as shown in Fig. 8, where we expect to lie with confidence 0.90 the CCDF curve compatible with the experimental data according to our model. This region is computed through a *peeling method*, as detailed in [3], for short, by discarding the most external curves up to a percentage of 10%.

4 Exploiting the Model

A second major technical problem to be solved in order to have an opportunistic network running is the way of transmitting a message. The general idea is that in an opportunistic network each radio device receiving a message is available to forward it to its neighboring devices. Each transmission from one device to the next is a hop. The hope is that with a limited number of hops each message reaches all its addressees. With this aim the naive flooding strategy – each device immediately forwards the received message to all neighboring devices – shows a certain number of drawbacks linked to: i) device power consumption, ii) backlog management of previously relayed messages that are still waiting to be delivered and could be outdated, iii) band saturation due to an intense replication of messages, and iv) collision between messages reaching a same device from different transmitters. Avoiding these drawbacks requires the setting up of some contrivances that may concern either a proper selection of the devices to which a message is passed [22], and/or the message timing [11], and in any case will provide rules for deciding when to stop the forwarding repetition of a given message. All these techniques require the involvement of sophisticated strategies, often endowed with probabilistic rules [15], generally based on statistics on the current status of the networks [46], e.g. the number of devices in the surroundings, the number of repetitions already done of the message, a list of IDs of the devices through which the message has already passed, and the like. The fact is that an optimization of the transmission protocol is hard even when the environment is known. On the other hand, as mentioned before, the basic elements of this environment described by the mobility models have not yet been completely assessed. Hence, from time to time we find segments of a protocol optimization theory based on specific hypotheses that definitely shed light on the problem, but require deep simulations and experimental campaigns to get truly operational results.

Our guess is that, on the one hand, non intensive communication networks which are moderately delay tolerant, like the one sketched in our example, may find *reasonable* solutions having the above drawbacks as *caveats*. On the other hand, we still need to assess some leading principles, such as the Pareto-like laws ruling intentionality, to have a robust vision of the phenomenon in a substantially distribution-free mode that will direct the results in more complex cases. As in any new operational field, these principles will be the offspring of former pioneering applications, where the latter take advantage of the methodology novelty when even broad results become satisfactory and rewarding.

5 Conclusions

In the previous century, scientists elaborating ideas around computing machines devoted primary attention to the program instructions, i.e. on how to process data. In contrast, in the new century the focus seems mainly on communication protocols, i.e. on how to communicate the data to be processed. Within this vein, a state-of-the-art template of communication paradigms is the Internet, where in essence we may appreciate that the most efficient communication protocol consists in the absence of social constraints, apart from some technological utilities. Hence no special permission or authentication is required in general, and on demand a user may typically receive thousands of records in response to an inquiry. The limitations of the web are obvious to everybody: a mass of data is available to any node of the network, with no guarantee of their value. Thus, except for skillful users capable of drawing the desired information, the vast majority of people are bombarded by a huge set of data, with very little time and limited skills for processing them. At its worst, the single user is affected by a set of mainly unstructured stimuli whose cumulative effect is not far from a Gaussian noise. To desaturate this information flow we may operate either from the bottom with feature selection methods such as random subspace [6, 24], or directly on the source of the data by selecting those of interest to us. Opportunistic networks are a tool for the latter option, where the selection derives from three factors: i) topological reasons, given the short range of the transponder devices; ii) transmission protocols, which are not based on the message content but on the transmission partners ID's; and, above all, iii) common intentions of the transmission partners, automatically giving rise to virtual communities at the basis of a collaborative computational intelligence.

In this paper we show, through the analysis of real-life intercontact times, the intentionality of the agents to be at the root of memory endowed processes forming the communities. From a statistical perspective this is due to the longer tails of the time distributions w.r.t. those of memoryless processes. Indeed, these tails allow us to relax the typical synchronization constraints that are at the root of the conventional parallel processing mechanisms. Rather, when the parallelism is exploited in terms of collaborative computational intelligence, the memory of the involved processes lets them take all the time that is necessary to meet the interested recipients of the messages to be processed.

References

1. Andrews, G.E., Askey, R., Roy, R.: Special functions (Encyclopedia of Mathematics and its Applications), vol. 71. Cambridge University Press, Cambridge (1999)
2. Algorithmic Inference, Wikipedia (2009), `http://en.wikipedia.org/wiki/Algorithmic_inference`
3. Apolloni, B., Bassis, S., Gaito, S., Malchiodi, D.: Appreciation of medical treatments by learning underlying functions with good confidence. Current Pharmaceutical Design 13(15), 1545–1570 (2007)
4. Apolloni, B., Malchiodi, D., Gaito, S.: Algorithmic Inference in Machine Learning, 2nd edn. Advanced Knowledge International, Magill, Adelaide (2006)
5. Brooks, F.P.: The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary edn. Addison-Wesley Professional, Reading (1995)
6. Bertoni, A., Folgieri, R., Valentini, G.: Bio-molecular cancer prediction with random subspace ensembles of support vector machines. Neurocomputing 63(C), 535–539 (2005)
7. Bhattacharjee, D., Rao, A., Shah, C., Shah, M., helmy, A.: Empirical modeling of campus-wide pedestrian mobility: Observations on the USC campus. In: Proceedings of the IEEE Vehicular Technology Conference, pp. 2887–2891 (2004)
8. Boudec, J.Y.L., Vojnovic, M.: Perfect simulation and stationarity of a class of mobility models. In: INFOCOM, pp. 2743–2754 (2005)
9. Camp, T., Boleng, J., Davies, V.: A survey of mobility models for Ad Hoc network research. Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications 2 (2002)
10. Chaintreau, A., Hui, P., Diot, C., Gass, R., Scott, J.: Impact of human mobility on opportunistic forwarding algorithms. IEEE Transactions on Mobile Computing 6(6), 606–620 (2007)
11. Chen, X., Murphy, A.: Enabling disconnected transitive communication in mobile ad hoc networks. In: Proceedings Workshop Principles of Mobile Computing, pp. 21–27 (2001)
12. Choffnes, D., Bustamante, F.: An integrated mobility and traffic model for vehicular wireless networks. In: Proc. of the 2nd ACM International Workshop on Vehicular Ad Hoc Networks (VANET) (2005)
13. Deng, J., Han, Y.S., Chen, P.N., Varshney, P.K.: Optimal transmission range for wireless ad hoc networks based on energy efficiency. IEEE Transactions on Communications 55(9) (2007)
14. Dorigo, M., Stutzle, T.: Ant Colony Optimization. Bradford Books. The MIT Press, Cambridge (2004)
15. Drabkin, V., Friedman, R., Kliot, G., Segal, M.: RAPID: Reliable probabilistic dissemination in wireless ad-hoc networks. In: SRDS 2007: 26th IEEE International Symposium on Reliable Distributed Systems, Beijing, China (2007)
16. Efron, B., Tibshirani, R.: An introduction to the Boostrap. Chapman and Hall/Freeman, New York (1993)
17. Einstein, A.: Investigations on the theory of the Brownian Movement. Dover Publication Ltd (1956)

18. Ferro, E., Potorti, F.: Bluetooth and wi-fi wireless protocols: A survey and a comparison. IEEE Wireless Communications 12(1), 12–26 (2005)
19. Fisher, M.A.: On the mathematical foundations of theoretical statistics. Philosophical Transactions of the Royal Society of London Ser. A 222, 309–368 (1925)
20. Georgii, H.O.: Gibbs measures and phase transitions. de Gruyter, Berlin (1988)
21. Gonzalez, M.C., Hidalgo, C.A., Barabasi, A.L.: Understanding individual human mobility patterns. Nature 453, 779–782 (2008)
22. Grossglauser, M., Tse, D.: Mobility increases the capacity of ad hoc wireless networks. IEEE/ACM Transaction on Networking 10(4), 477–486 (2002)
23. Gustafson, J.L.: Reevaluating amdahl's law. Communications of the ACM 31, 532–533 (1988)
24. Ho, T.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
25. Jain, R., Lelescu, D., Balakrishnan, M.: An empirical model for user registration patterns in a campus wireless lan. In: Proceedings of the Eleventh Annual International Conference on Mobile Computing and Networking (mobiCom), pp. 170–184 (2005)
26. Johnson, D., Maltz, D.: Dynamic source routing in Ad Hoc wireless networks. In: Imielinski, T., Korth, H.F. (eds.) Mobile Computing, pp. 153–181. Kluwer Academic Publishers, Dordrecht (1996)
27. Josang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems 43(2), 618–644 (2007)
28. Kangasharju, J., Heinemann, A.: Incentives for opportunistic networks. In: AINAW 2008: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops, pp. 1684–1689. IEEE Computer Society, Washington (2008)
29. Karagiannis, T., Le Boudec, J., Vojnovic, M.: Power law and exponential decay of inter contact times between mobile devices. Tech. rep., Microsoft Research, Cambrdidge, UK (2007)
30. Kim, M., Kotz, D., Kim, S.: Extracting a mobility model from real user traces. In: Proceedings of IEEE INFOCOM (2006)
31. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
32. Kurose, J.F., Ross, K.W.: Computer Networking – A Top-Down Approach, 4th edn. Pearson Addison Wesley, London (2008)
33. Lelescu, D., Kozat, U., Jain, R., Balakrishnan, M.: Model T++: An empirical joint space-time registration model. In: Proceedings of ACM MOBIHOC, pp. 61–72 (2006)
34. McCulloch, W., Pitts, W.: A logical calculus of ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5, 115–133 (1943)
35. Muqattash, A., Krunz, M.: CDMA-based MAC protocol for wireless ad hoc networks. In: MobiHoc 2003: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing, pp. 153–164. ACM, New York (2003)
36. Musolesi, M., Mascolo, C.: A community based mobility model for Ad Hoc network research. In: REALMAN 2006: Proceedings of the second international workshop on multi-hop Ad Hoc networks: from theory to reality, pp. 31–38. ACM Press, New York (2006), http://doi.acm.org/10.1145/1132983.1132990
37. Papoulis, A.: Probability, Random Variables, and Stochastic Processes, 2nd edn. McGraw-Hill, New York (1984)
38. Rohatgi, V.K.: An Introduction to Probablity Theory and Mathematical Statistics. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York (1976)

39. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323, 533–536 (1986)
40. Schuler, D.: Social computing. Communication of ACM 37(1), 28–29 (1994)
41. Stigler, S.: Studies in the history of probability and statistics. xxxii: Laplace, fisher and the discovery of the concept of sufficiency. Biometrika 60(3), 439–445 (1973)
42. Straub, T., Heinemann, A.: An anonymous bonus point system for mobile commerce based on word-of-mouth recommendation. In: Liebrock, L.M. (ed.) Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 766–773. ACM Press, New York (2004)
43. Surowiecki, J.: The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations. Little, Brown (2004)
44. Tapscott, D., Williams, A.D.: Wikinomics: How Mass Collaboration Changes Everything. Penguin Books Ltd, Lodon (2007)
45. Toh, C.K.: Ad Hoc Mobile Wireless Networks. Prentice Hall Publishers, Englewood Cliffs (2002)
46. Tseng, Y.C., Ni, S.Y., Chen, Y.S., Sheu, J.P.: The broadcast storm problem in a mobile ad hoc network. Wireless Networks 8(2/3), 153–167 (2002)
47. Wilks, S.S.: Mathematical Statistics. Wiley Publications in Statistics. John Wiley, New York (1962)
48. Yoon, J., Liu, M., Noble, B.: Random waypoint considered harmful. In: Proceedings of INFOCOM, IEEE (2003), http://citeseer.ist.psu.edu/yoon03random.html

Part VII
Artificial Immune Systems

Exploiting Collaborations in the Immune System: The Future of Artificial Immune Systems

Emma Hart, Chris McEwan, and Despina Davoudani

Abstract. Despite a steady increase in the application of algorithms inspired by the natural immune system to a variety of domains over the previous decade, we argue that the field of Artificial Immune Systems has yet to achieve its full potential. We suggest that two factors contribute to this; firstly, that the metaphor has been applied to insufficiently complex domains, and secondly, that isolated mechanisms that occur in the immune system have been used naïvely and out of context. We outline the properties of domains which may benefit from an immune approach and then describe a number of immune mechanisms and perspectives that are ripe for exploration from a computational perspective. In each of these mechanisms collaboration plays a key role. The concepts are illustrated using two exemplars of practical applications of selected mechanisms from the domains of machine learning and wireless sensor networks. The article suggests that exploiting the collaborations that occur between actors and signals in the immune system will lead to a new generation of engineered systems that are fit for purpose in the same way as their biological counterparts.

1 Introduction

The natural immune system, one of nature's most complex and fascinating systems, first provided inspiration for computer scientists in the 1990s. Since then, the rapidly evolving paradigm of Artificial Immune Systems (AIS) has developed, with a

Emma Hart
Napier University, Edinburgh
e-mail: e.hart@napier.ac.uk

Chris McEwan
Napier University, Edinburgh
e-mail: c.mcewan@napier.ac.uk

Despina Davoudani
Napier University, Edinburgh
e-mail: d.davoudani@napier.ac.uk

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 527–558.
springerlink.com

thriving community of researchers world-wide. A glance through the dedicated AIS conference series ICARIS [2], or through the many papers now published in a diverse array of journals highlights the variety apparent in both the domains that AIS algorithms have been applied to, and in the design of the algorithms themselves. Unlike in other biologically inspired areas of computation (for example Evolutionary Computing), there is no canonical AIS algorithm[1] — the diverse array of immune-inspired algorithms have roots in an equally diverse set of immune mechanisms. The various algorithms are described in detail in a number of publications - the reader is referred to a textbook such as [19] for an introductory overview or to recent journal publications such as [50] or [28] which discuss the current state-of-the-art in detail. The purpose of the article is not to recover this ground, but to examine some overlooked aspects of the natural immune system which we believe provide a richer (and currently unexploited) source of inspiration for computational systems in the future than current algorithms; in particular, we place significant emphasis on the design of *systems* rather than *algorithms*. The article suggests some possible reasons why the potential promised by AIS has not been delivered, and suggests how this might be addressed in the future. The arguments are particularly relevant in light of recent advances in technology which present a new and challenging range of problems to be solved.

1.1 A Reflection on AIS Today

Analysis of the AIS literature reveals a number of perhaps surprising facts. Firstly, computational algorithms tend to exploit individual, naïve mechanisms that are apparent in the immune system, despite the obvious fact that these mechanisms do not operate in isolation in the natural immune system. Thus, we observe algorithms derived from mechanisms such as clonal selection [8], negative selection, and immune networks [32]. Algorithms derived from clonal selection principles have been applied in the *optimisation* [11] and *classification* [55] domains. Negative selection acts as inspiration for a number of *anomaly detection* algorithms, leading on from seminal work in the AIS field from Forrest *et al* [22, 21] (though note that there is an increase in recent literature which questions the theoretical basis of these algorithms as anomaly detectors [47] due to questions of scalability). Algorithms derived from immune network theory are prevalent particularly in *robotics* [56] but also in continuous learning algorithms which perform clustering [36]. These problem domains are familiar to many branches of computational learning, and have been addressed by many other *non-AIS* approaches and techniques. Immune-inspired algorithms have provided useful alternative perspectives to many of these more traditional techniques, and have often been shown to achieve comparable, or sometimes better results on some problems than some other algorithms. However, it is unclear what the immune metaphor, applied in these contexts, can offer over and above that which

[1] Note that the authors hold the position that this is in fact beneficial.

can be already achieved by other more traditional methods [2]. To this end, we argue that application of immune inspiration to these problem domains is failing to exploit the true potential of the metaphor of the immune system as a complex system.

Secondly, research and therefore algorithm development has increasingly become driven by engineering requirements; algorithms are developed to solve a particular problem, and then tweaked and tuned to solve the particular problem instance at hand. This prevents the development of *principled* algorithms which can be generalised to other domains. This trend goes side-by-side with an increasing tendency for algorithms to move further and further away from the biology that originally inspired them. This is often referred to by Stepney *et al* [44] as "reasoning by metaphor" - Timmis argues further in [50] that the limited perspective of the immune system exploited in current AIS algorithms will ultimately limit the success of the field. On a more positive note, it is worth noting that this trend *is* beginning to be reversed - recent publications such as [48, 34, 25] present new, inter-disciplinary viewpoints which have roots firmly in the underlying immunology.

Thirdly, and perhaps underlying both of the previous points however has been a tendency to attempt to exploit what is commonly regarded as *classical* immunology. This refers to the common assumption that the immune system's major function is to separate *self* from *non-self*, where *self* defines the 'normal' state of the immune system host and *non-self* everything else. The point that this is a general perception is perhaps made best with reference to the Wikipedia definition of the immune system *"The immune system defends the body by recognizing agents that represent self and those that represent non-self, and launching attacks against harmful members of the latter group"*. It does not seems surprising therefore that this viewpoint often results in the perception in those working outside of the immediate field that AIS algorithms are associated mostly with domains such as security and anomaly detection, as a form of defence.

However, despite the ubiquity of the self/non-self theory of immunology, a number of alternative camps exist in the immunological world, each with sizeable followings, and proposing a number of different theories which question not only the mechanisms by which the immune system is held to operate but more fundamentally, the actual role of the immune system itself. Such theories are not necessarily mutually exclusive to classical immunology, rather they present a different perspective on both the functioning and the role of the immune system which has significant potential for engineering systems. For example, while host defence is clearly a critical function, [12, 45] have proposed that it cannot be the only function of interest. In a radical departure from accepted thinking, recent work by leading immunologist Irun Cohen has suggested that the immune system plays a much more fundamental role in the body than simply protecting it from harm, and instead, its function is that of *body maintenance*. Although this may seem a subtle semantic point, in this view (expounded in detail by Cohen in [13]), detection of harmful situations becomes merely a special case of overall body maintenance. The immune system

[2] Furthermore, many alternative, perhaps older fields, offer algorithms which are backed up by theory (which tends to be currently lacking in AIS), and hence are more readily accepted.

instead is seen as a *cognitive system* (in the sense that is computes state and acts upon that state) which continuously provides body maintenance to the host. In the body, the term maintenance covers a wide and diverse spectrum of functions, ranging from healing of cuts and bruises, to inflammation, to mending broken bones. In [14] Cohen likens the functioning of the immune system to that of a computational machine, in that it must compute its current state and act upon that. His suggestion that by computing both its internal and external state the immune system can be re-configurable, adaptive, secure, and provide self-healing functionality in an unpredictable and dynamic environment is clearly appealing from an engineering context.

Regarding the immune system as a system which provides maintenance rather than defence considerably widens the scope of engineered applications which might benefit from an immune inspired approach. Additionally, the properties just mentioned suggest application of immune-inspired mechanisms to a far richer field than is currently apparent. The motivation of this article therefore is as follows. First, we ground the discussion by focusing on the features and properties of challenging engineered systems which necessitate novel computational approaches. We then describe a number of immune-mechanisms which are yet to be fully exploited in computational systems. The discussion re-positions the immune system as a complex, collaborative system of multiple signals and actors. A number of examples of systems in which steps are currently being taken to implement some of the mechanisms are then described. We conclude with a discussion of an emerging field, that of *immuno-engineering* which promises a methodology which will facilitate maximum exploitation of immune mechanisms in the future.

1.2 Challenges Posed by Real Systems

Computational devices are now ubiquitous — in every home, in every office, on every street, numerous small and inexpensive devices are becoming capable of spontaneously networking to each other and to the Internet, bringing with this ability the potential of a new age in computing which is data, rather than technology, driven. We can now construct autonomous systems, ranging from perhaps a robot containing tens of simple devices, to mobile, adhoc networks containing 1000s of such devices. At both extremes, such systems consist of unreliable heterogeneous sensors and actuators, which must make decisions across multiple time-scales in unpredictable, and potentially hostile, dynamic environments in order to maintain their integrity and achieve their desired functionality [49]. Current technology allows us to hard-wire responses to foreseeable situations; a considerable void is still to be crossed however to achieve systems which adapt continuously and autonomously to their environments and exhibit what is becoming known as self-CHOP characteristics; self-configure, self-heal, self-optimize and self-protect. A paradigm shift in engineering is required to address this — current methodologies in software engineering will not suffice when attempting to develop networks that are self-managing, self-diagnostic, and above all, robust and secure in ever changing dynamic and unpredictable environments. This view is reinforced by Zambonelli and Panurak who note that *the complexity introduced to software systems by several emerging computing scenarios goes beyond the capabilities of traditional computer science and software engineering abstractions* [59]. We propose that when attempting to address such challenges in building and maintaining truly complex systems, then the immune system metaphor is ripe for exploration [27].

The properties we wish to endow on complex engineered systems are exhibited by the immune systems of many complex living systems. Such systems possess an immune system which comprises of an innate component which endows the host with rapid, pre-programmed responses and an adaptive component which is capable of learning through experience. Much of the desired functionality of the system arises from the interplay between these subsystems and the regulatory effect they have on each other. Together, these operate over multiple time-scales, from seconds to the entire lifetime of the organism, endowing a *system* with the ability to function, and maintain itself over its lifetime. Application of immune-inspired mechanisms to engineered *systems* which are required to exhibit the same kind of properties will need to be in stark contrast to the type 'traditional' AIS algorithm development prevalent in the literature. For example, many efforts have been made to derive *optimisation* or *classification* algorithms by looking to a natural system which arguably does not exhibit the hoped for functionality at all (optimisation) or where the functionality is one smaller part of a bigger picture (classification). The features of applications which are likely to profit from this area are summarised in previous research by one of the authors in collaboration with Timmis [28], and are listed below. We stress that the biggest rewards are likely to be observed in applications which exhibit *all* of these features — systems which exhibit only one or two features from the list are unlikely to profit fully.

1. They will be embodied
2. They will exhibit homeostasis
3. They will benefit from interactions between the innate and adaptive immune systems
4. They will consist of multiple, heterogeneous interacting components
5. They will be easily and naturally distributed
6. They will be required to perform life-long learning

The properties above are mainly self-explanatory, with perhaps the exception of the first two: *Embodiment* from the point of view of the type of systems we describe is perhaps best understood in terms described succinctly by Stepney in [43]. A system which is *embodied* in its environment is a system which can sense and manipulate its own environment, with its internal state depending on what it senses, and its actions depending on its state. The system's actions may change the environment which affects what it subsequently senses ... and thus its subsequent actions, thus producing a complex dynamical feedback. *Homeostasis* in a system generally describes the ability of an organism to regulate its internal environment such that it remains in a stable and constant condition and has a natural mapping to an engineered system which must maintain itself in some *viability zone* [7] in order to operate successfully.

Neal and Trapnell provide further elaboration on point (4) above; in [37] they discuss in detail the numerous actors and interactions in the immune system. The arguments presented in [28] are extended, stating that appropriate exploitation of immune collaborations will have the following benefits:

- Systems requiring multiple representations of actors, information and interactions should become tractable and appropriate
- Systems requiring distributed detection of correlations of events and/or patterns can be considered
- Systems requiring close integration and distribution of pattern recognition mechanisms with existing complex engineered systems can be considered.

This is a compelling view. Therefore, in the remainder of the article we first provide a brief overview of some of the mechanisms that are observed in the natural immune system in which collaboration plays a crucial role, and in which we believe provide much inspiration for the computer scientists solving complex problems in the future. The material covers two perspectives. Firstly, we consider mechanisms apparent in the natural immune system which are currently unexploited in artificial systems. Secondly, we discuss recent work in immunology which attempts to reposition the immune system away from a pure defence mechanism to a complex, self-organising computational system. Finally, we suggest some future directions for AIS research; complex problems require complex solutions — paying closer attention to the intricacies of the immune systems rather than resorting to "reasoning by metaphor" can potentially reap rich rewards.

2 The Natural Immune System

The natural immune system is one of the most complex systems in nature, consisting of numerous players and mechanisms. Research in immunology flourishes worldwide - in 1970, Jerne [32] estimated that the number of immunologists in the world had tripled every 20 years since the late 19th century, and Coutinho[15] recently estimates the current number as around 40000, with a new paper in immunology published on average every 15 minutes. Despite this, many of its mysteries remains unsolved, and a particular paradox of the field is that despite huge advances in basis science, there has been few clinical applications resulting from this [15]. There are several competing (though not necessarily mutually exclusive) theories as to how immune function is achieved, each associated with a plethora of mechanisms. Although unsatisfactory from a clinical point of view, this is not necessarily a hindrance for the computer scientist — indeed, one might argue that exactly the opposite is true. Unencumbered by the need to explain experimentally observed data, the computer scientist is at liberty to pick and choose from the theories of the immune system described in the literature [3].

[3] The computer scientist does not have a completely free hand; as described in Section 6 and advocated by [44] immune-inspired algorithms should result from careful abstraction of natural mechanisms and be under-pinned by theory.

We present a brief overview of some of the fundamental processes that occur in the natural immune system. Our overview omits much detail, and in particular does not cover the 'classical' theories of clonal selection and negative selection in the adaptive natural immune system which has inspired much of the work in the AIS field. Background on these processes can be found in the introduction to almost any AIS paper. Additionally, the interested reader is referred to accessible immunological texts such as [42] and [31] or to [19] for a computer-scientists' perspective. A very detailed discussion of the many actors and interactions apparent in the natural immune system is given by Neal and Trapnell in [37] - this text gives a much more detailed account of vast numbers of cells and interactions occurring in the immune system than is provided here, and draws parallels with the type of engineered system that could benefit from such interactions. We focus particularly on some of the immunological mechanisms which particularly rely on collaboration. Some of the these mechanisms have been overlooked by the AIS community, particularly those occurring in the innate immune system which has often been overlooked as being too simplistic by researchers in computational intelligence. Hence, they are presented here to stimulate discussion and future research. Others, such as the cognitive immune theory of Cohen, have attracted much attention recently in the AIS community. Timmis and Andrews proved a thoughtful theoretical perspective on the usefulness of this theory in [48, 5, 3]; work of a more practical flavour which exploits Cohen's ideas is discussed more detail later in this chapter in Sections 4.1 and 5. The collaborative theme is evident throughout all the mechanisms reviewed, occurring in the innate IS, between the innate and adaptive systems, and in the adaptive system itself.

2.1 Innate vs. Adaptive Immunology

The natural immune system consists of several layers or subsystems. A first line of defence is provided by physical barriers such as the skin, but beyond this, lie two interacting complex systems, the *innate* immune system and the *adaptive* immune system. The innate system is present in all animals, and provides a number of programmed responses to invading bacteria and potentially harmful signals. It is the innate system that is responsible amongst other things for *inflammation*, one of the first outward signs that an immune response is occurring, and for sending signals to alert the brain that something hurts! 99% of animal species exist with only an innate immune system. In vertebrates however, another level of defence, the adaptive system, provides an extra layer of response which can adapt dynamically to any invader which is causing harm. The latter system has attracted much attention in AIS whilst the former has been largely ignored. A number of salient innate mechanisms are now described.

2.2 Cooperative Innate Immunology

The innate immune system includes a number of important cells such as the complement system, professional phagocytes, and natural killer cells. We discuss the role of the latter two players here; a further key innate player, the dendritic cell, is discussed in more detail in the next section. The system works efficiently due to the *cooperation* of the various cells; only via a collaborative effort can the innate system respond quickly and strongly when under attack. The role of some of the actors and the mechanisms by which they cooperate are now described.

2.2.1 Macrophages

The "professional" phagocytes of the immune system have a primary function of phagocytosing or eating other cells. The most important of these cells is the macrophage, which plays a number of roles which are of potential interest; in their resting state, they function as a garbage collector, clearing cells of debris. However, if they receive environmental signals which indicate that the initial defence barriers have been penetrated, they change function: they increase the rate at which they engulf other cells, and thus begin to engulf invaders. At the same time, the *information* that describes the engulfed invaders (i.e. the proteins) is displayed on the surface of the macrophage. The activating signals come from a class of immune *communication* molecules known as cytokines — the best studied is a cytokine called interferon-gamma, INF-γ. Macrophages can exist in a third state which arises via direct activation as a result of signals received directly from invading cells and is known as hyperactivaton. In this state, the signals received from invading cells (such as LPS) cause the macrophage to switch behaviours - it stops proliferating, increases in size, and becomes a 'killer', phagocytosing invaders. Additionally, the macrophages release other signalling molecules (in particular tumor necrosis factor TNF) which can both act as killers themselves *and* recruit other collaborators in the immune system. The various signalling mechanisms provide important guidance to the innate immune system, and also emphasise the role played by the environment. Environmental signals activate a set of *behaviours* in the immune system and also recruit other players into cooperating.

2.2.2 Natural Killer Cells

Anther important class of cells is the family of cells known as Natural Killer Cells (NK cells). These cells kill invaders by forcing them to commit suicide. NK cells appear to directly target invaders – a two signal exchange between a potential target cell and the NK is thought to determine whether or not the target is killed. Furthermore, NK cells also release the activating cytokine IFN-γ. As with macrophages, NK cells require activating signals from the environment; several signals, produced only when the body is under attack, have been identified.

2.2.3 Cooperation

The clever cooperation between these cells which results in an efficient system is shown in Figure 1. For example, signals from an invading bacterium activate the NK cell; this responds by producing INF-γ, the signal that activates macrophages.

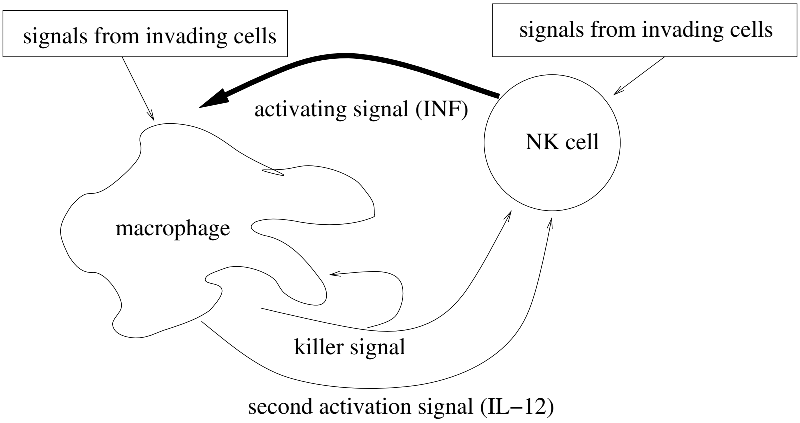

signals from invading cells
signals from invading cells
activating signal (INF)
NK cell
macrophage
killer signal
second activation signal (IL−12)

Fig. 1 Cooperation between macrophages and natural killer cells provides a self-regulatory system, adapted from [42]

The activated macrophage becomes hyper-activated when it receives this signal *and* the signal from the invading bacterium. The hyper-activated macrophage produces the killer cytokine TNF. This cytokine is detected by the macrophage itself which causes it to release another molecule IL-2. Together, the released IL-12 and the TNF activate NK cells which produce more INF-γ..... and a positive feedback loop is set up which in turn primes more macrophages. The cooperation between the different classes of cells play a dual role; it provides reinforcement via positive feedback loops and also provides confirmation to cells that their diagnosis of a situation is correct.

Currently, there exist very few applications in AIS which attempt to exploit the immunological features described in Sections 2.2.1 to 2.2.3. One the one hand, the innate immune system is inherently more simple than the adaptive immune system; yet, until very recently, almost all focus in the computational literature was on the adaptive system. Secondly, many natural systems function perfectly well with only an innate immune system - perhaps computer scientists ought to look to the innate immune system in the first place in order to design artificial systems. For example, it seems clear that progress could be made in a number of robotics and control applications (particularly those concerned with fault tolerance) by exploiting the type of feedback loops exemplified in the immune system just described, without having to resort to more complex systems at all.

2.3 Dendritic Cells

Until very recently, it was thought that the only role of the innate system was to provide an indiscriminate, rapid defence until an adaptive response kicked in; however, it is now clear that the innate system is actually responsible for sensing danger and then activating the adaptive system. Furthermore, rather than simply turning on the adaptive system, the innate system essentially computes the state of the body, and returns this state to the adaptive system. The state information returned to the adaptive system includes information regarding the type and location of the attack, thus imparting knowledge on how and where to react. This is achieved via dendritic cells, which scout the body tissues to determine its state, integrate the information, and then signal to the adaptive system whether to react or not.

For this reason, the dendritic cell is often referred to as the ‘sentinel’ of the immune system [42]. Dendritic cells reside in the epithelial tissues of the body (for example the skin). These cells migrate through the tissue, sampling the tissue in their vicinity. Essentially, the dendritic cells soak up molecular debris (for example, bacteria or other pathogenic material) and additionally, sense molecular signals present in the tissue. Some of these signals derive from safe or normal events such as regular, or pre-programmed cell-death (apoptosis). Other signals are derived from potentially dangerous events — for example, an exogenous signal known as a PAMP is produced exclusively by pathogens. Another class of endogenous signals known as danger signals are produced by by cells which die as a result of stress or from attack. Exposure to sufficient levels of either signal results in the dendritic cell maturing into one of two states, known as *semi-mature* or *mature*. The matured cell then migrates back to the nearest lymph node through a complex system of lymphatic vessels via a process known as chemotaxis.

The lymph nodes function as molecular dating agencies where the different immune cells of the body congregate — their small volume increases the probability of cellular interactions. In particular, the dendritic cells that reach the lymph node carry a snapshot of the current state of the tissues back. The snapshot contains two important pieces of information: antigen, i.e. material causing a problem, and also the signals representing the *context* under which the material was collected. This snapshot is viewed by the reactive immune cells, in particular T-Cells, and a process of communication, and collaboration between cells ensues which ultimately results in activation or tolerance of the immune system, depending on the content and context of the information presented. Returning DCs which have been exposed to antigenic material in the context of *safe* signals are known as semi-mature; these induce tolerance in the T-cells present in the lymph node. Those DCs which have returned having been exposed to antigen in the context of PAMP and Danger signals induce a reactive response. This is shown in Figure 2. Thus, antigens are no longer considered dangerous *per se*, it is their context and resulting effect on the environment that determine their ultimate outcome.

This aspect of dendritic cell behaviour has inspired computational research concerned with anomaly detection, in which the basic problem is to determine whether to ignore or react to patterns of information in a system (for example, in a computer). This approach shifts the focus of a detection algorithm to understanding the effects of an intrusion to a system, rather than the signature of the intrusion itself. This has perhaps significant advantages in that the effects on a system are potentially easier to measure than patterns of incoming information which may be numerous and diverse in their nature e.g. [25]. Furthermore, the dendritic-cell-algorithm (DCA) of Greensmith *et al* encapsulates a *time* dependent method of reacting to the effects of

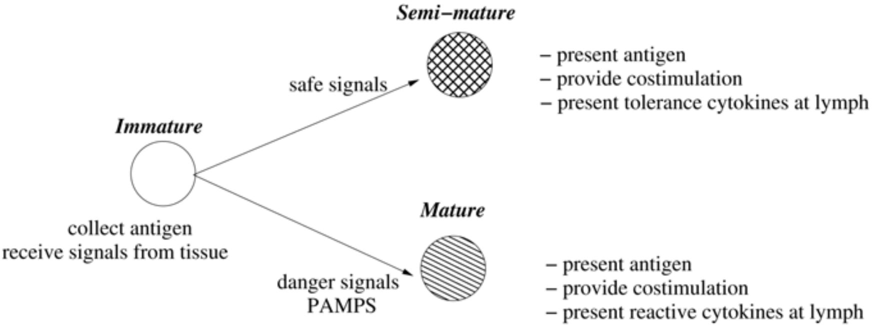


Fig. 2 The pathways of dendritic cell differentiation are dependent on the amount and type of signal received. Mature cells eventually activate an immune response in the lymph node. Semi-mature cells collect antigen, i.e. non-self material, but do not activate a response to this material

an accumulation of signal over time, rather than reacting to individual incoming patterns or signatures. The dendritic system has also inspired applications in the broad area of wireless sensor networks [17, 16, 18]. These applications exploit two essential properties of this system, the first concerned with the action of the dendritic cells themselves and the second with the lymphatic dating agency just discussed. The former mechanism exploits the notion of DCs as mobile agents which scout an environment, collecting information about the contents of that environment. The latter is concerned with how the collective information returned by numbers of DCs to the lymph node is integrated and interpreted and ultimately results in a system wide response. We return to this in more detail later in the chapter in Section 3.1 in a discussion of the *cognitive* immune system.

3 The Adaptive Immune System: Carneiro's Networks

The majority of existing AIS applications are based around interpretation of adaptive immune mechanisms (e.g clonal selection). The major players in these mechanisms are the B-Cell and T-Cell. Immunologists today take for granted that T-cells and B-cells must interact to realize the full potential of the immune system, but it is telling that this fact has to date been overlooked in AIS. In fact, many of the successful AIS algorithms which have been derived from adaptive immune mechanisms (e.g aiNET [11] and others discussed in Section 1.1) focus on modelling only B-cell interactions, ignoring T-cell collaboration altogether[4]. Here, we briefly describe the role of these two cells before describing in detail the immunological basis of a model which exploits the interactions between the cells to great effect. Later in the chapter, we describe a computational interpretation of this model and suggest possible applications.

[4] An exception is the work on network intrusion detection using the LYSIS system of[29] which did incorporate the requirement of a *second signal* for cells to become activated.

B-Cells are white blood cells produced by the bone-marrow that mature to produce antibodies. Each antibody can bind to a specific set of antigens according to its molecular shape, with the specific antigen to which it binds referred to as its *cognate* antigen. This process gives rise to the popular "lock and key" metaphor of immunology, describing the process by which antibodies (keys) and bind to specific antigens (locks). Most AIS models adopt a simplified view of the immunology, exploiting the fact that ultimately, recognition of a cognate antigen by a B-cell causes it to proliferate, producing clones which have receptors which recognise the same antigen - a process known as clonal selection. However, the process is actually more complex:

T-Cells are similar to B-Cells in appearance, and display on their surface antibody-like receptors known as TCRs, *T-cell receptors*. Like B-cells, these receptors also bind to their cognate antigen, however the process has a number of crucial differences. Firstly, T-Cells only recognise protein antigen, unlike B-cells which can recognise any organic molecules. Most importantly however, T-Cells can only recognise antigen that is presented to them displayed on the surface of a further class of molecule known as MHC, in contrast to B-cells which require no help to recognise their cognate antigens. MHC processes antigen into peptide fragments, which are displayed on the MHC surface. T-cells recognise short linear chains of these peptides which correspond to contiguous sequences in the primary structure of the antigen itself. B-cell binding on the other hand involves direct binding of the BCR to the antigen, which therefore recognise the secondary, or *surface* structure of the antigen, i.e. amino-acids that are discontinuous in the primary structure but are brought together in the folded protein. These differences in the *level* of abstraction at which recognition occurs play an important role in the collaborative response of these cells, and one which we exploit in a machine learning context as explained later in Section 4.1.

Recognition of cognate antigen by T-Cells causes them to become activated, at which point they secrete cytokines. These cytokines produced by the T-Cell provide a crucial *second signal* to a B-Cell which has also recognised a cognate antigen, confirming to the B-Cell that it should become activated. Without this signal, B-Cells which have recognised cognate antigen through their own receptors (BCRs) do not become activated. Thus, the interaction between the T-Cell and B-Cell is critical in turning on the immune response.

In theoretical immunology, many of the seminal models (typically described as coupled differential equations) are concerned with the dynamics of the *clonal selection* process – how receptors bind ligands and then induce proliferation, secretion and mutation of antibodies in the immune repertoire. This is extended in idiotypic network theories [32] by incorporating the ability of antibodies to bind both antigen *and* other antibodies, but the fundamental processes in the models remain similar. In both cases, inter-clonal competition, i.e the relative ability of clones to proliferate, is a function of fitness in ability to bind ligands and thus be selected, but there is no real sense of *co-operation* amongst clones.

A lineage of work in theoretical immunology (starting notably from Jerne [32] and continuing through to work by Carneiro [9]) has evolved these models to

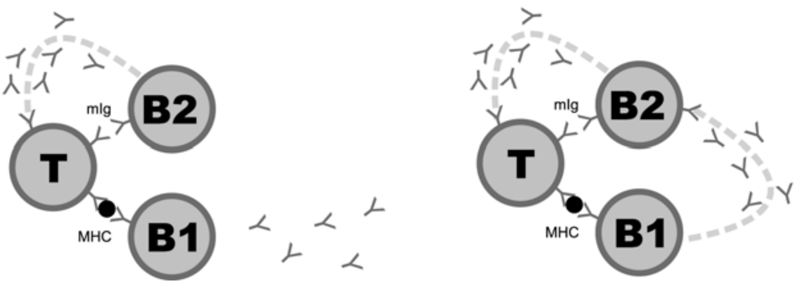


Fig. 3 **Left:** Two modes of B-T co-operation (*i*) B1 achieves the sustained surface proximity to T necessary for T-Help via MHC-Presentation. By virtue of T-Cell receptor/B-Cell receptor (TCR/BCR) morphological similarity, the resulting secreted antibody has little affinity with the TCR and both clones proliferate in an explosive positive feedback loop. (*ii*) B2 interacts via direct BCR recognition of the TCR. The resulting secreted antibody thus has affinity and directly suppresses T in a negative feedback loop. **Right:** an illustration of how these complementary interactions "close the loop" and drive each other into a stable configuration via idiotypic interactions between B-Clones

demonstrate compelling behaviour of emergent tolerance and immunity driven by endogenous criteria — exactly the type of behaviours which have encouraged computer scientists and engineers to attempt to induce the similar behaviours in engineered systems. However, although these models show that it was possible to partition a space into tolerant and immune regions, thus capturing the the *function* of the immune response, they failed to capture the *structure* of such a response, particularly in regard to the ability of the models to maintain a distinction between *self* (that which should not be reacted to) and *non-self*, (that which should be reacted to). This failure results from an implicit requirement for *symmetry* in a space imposed by the lock and key notion which requires that complementary shapes interact. This is of critical importance in both immunology and computational or engineering applications: imposing a complementary shape-space on a model or in an algorithm has fundamental implications for the resulting *structure* and therefore *functionality* of a space[5].

Carneiro *et al* [9, 10] however extended the existing models (which are generally presented in a mathematical form as a series of differential equations [40]) and showed that it was possible to break this underlying requirement for symmetry by integrating the co-operation between B-Cells and T-Cells in regulating the adaptive immune response [46].

Although the integration of T-Cells is a conceptually straight-forward modification to existing models, its derivation and consequences are quite profound. We refer the reader to [9, 10] for the original details and analysis and instead, attempt to decouple the significant mechanisms of the model from the overall dynamics (which later in the chapter are interpreted from a statistical learning perspective). The central notion is that B-Cells cannot achieve activation without *a second signal from*

[5] See [26] for a detailed discussion of the relationship between structure and function in a number of geometric spaces.

co-operative interaction with T-Helper cells. The two modes of possible interaction are illustrated in Figure 3 where each induce positive and negative feedbacks loops, respectively.

B-Clone activation (and thus antibody secretion) is limited by available T-Help; T-Help, in turn, is limited by the suppressive effects of TCR-specific antibody on T-Clones. The complementary waves of immune response initiated by the antigen (then by antibody, then *anti*-antibody etc) thus self-regulates, with ultimate tolerance or immunity resulting from the *indirect* competition between T-Clones and antigen to survive suppression from antibody. As B-Clones compete to garner available T-Help, only the fittest achieve activation and secretion, relegating weaker clones out of the repertoire and focusing it to best reflect the antigenic environment of the host.

Thus there are several forms of asymmetry in the model: (*i*) T-Clones act as a driving force, limiting factor and target of the immune response, much like their cognate antigen; however T-Help is provided to B-Clones via a separate mechanism that is crucially, independent of the "shape" of the BCR (*ii*) TCR-specific B-Clones have an inherent advantage in binding T-Clones, and thus receiving T-Help, because their binding is based on a BCR-TCR rather than an MHC-TCR pathway and there are many more BCR on a B-Cells surface (*iii*) Complementary B-Clones can only sustain traditional oscillatory relationships while both antigen and T-Clone survive; the eradication of either T-Clone (tolerance) or antigen (immunity) leads to a collapse of the repertoire where one response dominates and the other is suppressed, and (*iv*) T-Clones only bind to single points in the shape-space, whereas B-Clones can bind to several, non-contiguous points (to simulate protein surface binding where distant peptides on the proteins primary structure are brought together on the surface of the folded secondary structure).

The latter two asymmetries are described in detail in [34], and are motivated by the wish to extend Carneiro's model beyond its original context [35] in order to apply it in a machine-learning context, however, the principles are still relevant in an immunological context. Figure 4 illustrates the mechanism in the case of a tolerance-inducing response to a high-dose antigen. The TCR-specific response is able to eradicate T-Clones before the antigen-specific response can eradicate antigen. The absence of TCR leaves the tolerance-inducing B-Clone in a slow decaying resting state. The immunity-inducing B-Clone, still stimulated by the now tolerated antigen, transitions into an induced state where lack of available T-Help forces the clone into a fast decaying *anergetic* death. The situation for immunity is equivalent but reversed. In both case, the difference between slow and fast decay ensures that the immune system "remembers" the correct response for an extended period and rapidly "forgets" the wrong response.

From a computational perspective, there are a number of benefits in being able to produce such behaviours in an engineered system. Later (Section 4.1), we describe how this model can be interpreted in a machine learning context, in a document classification or query expansion application. In this type of application, a system must *learn* how to respond to incoming data, for example, information contained in a stream of documents. In such a scenario, the system might respond to *words* in the documents, depending on the context in which they are presented and

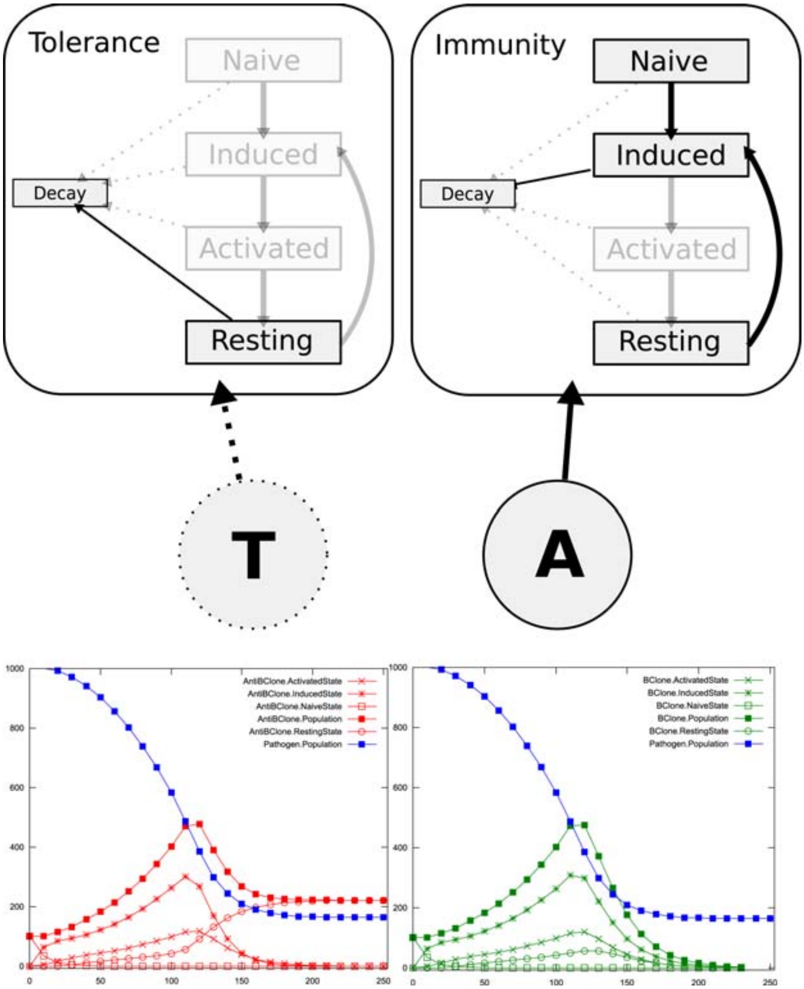


Fig. 4 Left: A schematic illustration of how eradication of either T-Clone or antigen (in this case, T-Clone) causes different state changes in immunity and tolerance inducing B-Clones that lead to fast decay (forgetting) and slow decay (memory). In both cases the loss of available T-Help from B-T co-operation is the trigger for rapid decay. **Right:** A graph of the dynamics of competing tolerance-inducing and immunity-inducing responses to a high-dose antigen. In this case tolerance out-competes immunity and the dominant response transitions into a slow decaying memory state while the recessive response is quickly eliminated

the previous history of presentation. The concepts governing the B/T cell interactions provide a possible mechanism for this by which response to any word (or other feature in a different domain) is a dynamic one governed by the competing B/T responses. The balance of the competing response ultimately results in a winner which determines whether tolerance of immunity to a feature is observed. Crucially for AIS, this mechanism does not require any *a priori* knowledge of good/bad, interesting/not-interesting etc. data — this is determined solely by context. The particular relevance of Carneiro's model is that it is unclear how these behaviours could be induced without modelling fine-grained co-operative interactions between heterogeneous cell types such as B and T cells: the loss of co-operative interactions is the trigger for rapid post-response correction.

3.1 The Cognitive Immune System

In a radical departure from classical immunology, Irun Cohen has popularised the *cognitive* immune system [13], viewing the immune system from a holistic perspective as a complex, adaptive and reactive system which is capable of *cognition*. This is a fundamentally different viewpoint to the classical immunology of the previous decade, and one that has attracted much attention in recent years in the AIS community, as it implicitly captures many of the properties one would like to mimic in intelligent systems. Cohen's arguments are compelling, but in a sense, represent more of a perspective than a tangible model; there are few quantitative models of the underlying immunology that can be directly exploited, though a notable exception to this is recent work by Voigt *et al* which captures the essence of the cognitive model in the framework of a learning classifier system [54]. Nonetheless, the concepts captured by the viewpoint offer a rich source of inspiration for the computer scientist or engineer. The arguments are elaborated below.

In line with the earlier network based approaches of Jerne and Varela [32, 51], Cohen proposes that the primary function of the immune system is *maintenance*. This may take the form of healing and repair (such as mending broken bones) as discussed in Section 1.1 or even defence. Neuman sums this up neatly in [38] *"Rather than promoting the metaphor of the immune system as a warrior that defends his castle against invaders.... the immune system is the maintenance man of the compartment building we call the organism"*. Such functionality demands far more complexity from a system than defence, requiring examining the cells and interactions that comprise the immune system in a completely different light. Cohen views the immune cells as forming an immune dialogue with the body tissues in which the condition is assessed via a collaborative effort from both adaptive and innate immune agents, which form a picture of both endogenous and exogenous activity (i.e of the body-self and of pathogens respectively).

Cohen re-frames the entire problem of attempting to define the proposed goals of immunity by considering the entire immune system as a *computational* system. The computational question is then not " which cells and interactions comprise the immune system" but rather "what is the immune system computing", which can only be answered by considering the *state* of the immune system [14].

Considering the immune system as cognitive, i.e that it computes states and effects actions given that state, may at first glance seem surprising. Cognition is usually thought of as a conscious process performed by the brain, but Cohen defines three elements that, when integrated, can result in a cognitive system without consciousness; the first requires a system to be capable of making *decisions* based on a number of potential choices. The second states that in order to accomplish this, the system must possess *internal-images* of its environment, which are updated based on its experience. The third element relates to the fact that the internal images are gained via interactions or *self-organisation*. In summary, update of the internal-images implies an increase of information which is driven through inputs of energy and information by the world, and similar outputs generated by the system. Eventually, these interactions result in choices, and therefore the emergence of cognition.

Cohen describes three important mechanisms which contribute to the emergence of cognition; *co-respondence*, *pleiotropia* and *redundancy*. The *co-respondence* concept suggests that in order to fulfill its role (maintenance, protection), the immune system maintains different classes of immune cells. These cells individually see different aspects of any object that may be of immune interest, from within the body (tissues) or external (antigens). Each class of immune cell informs other immune cells about what it has seen, by expressing co-response signals (cytokines, processed peptides, interaction molecules, antibodies). The effect of these signals, essentially, is that each cell modifies its own response based on the feedback it receives from the other cells [13] — essentially, although it is impossible for a cell to perceive what another cell perceives of its environment, it can perceive how another cells responds, and therefore respond to this response. Through this cooperative response and dialogue, the immune system can generate a system-wide view of an event and its context.

Pleiotropia denotes the capacity of a single immune component to produce several diverse effects. Depending on existing conditions, immune agents elicit different responses and do different, sometimes contradictory things. For example, in the natural immune system, a T cell can kill a target cell and stimulate the growth of another. This is another essential property for components of an autonomic system, in which components must be capable of effecting a number of different responses depending on current environmental conditions.

The final property, *redundancy* is distinguished as simple and degenerate. Simple redundancy designates the existence of multiple copies of the same element (e.g. numerous antibodies are produced during an immune response), while degenerate redundancy describes the situation in which many different immune components perform the same action. Both types are relevant to large, distributed networks containing multiple components. Assuming unreliability of individual network elements , multiple copies of the same *network component* will inevitably exist in any application, and in addition, different components may respond in a similar manner to certain environmental conditions.

This is a short summary of the Cognitive Model — the immunological basis is described in detail in [13] and summarised by Andrews in [4] for the computing community. Perhaps however the theory (and its potential use) is summed up most succinctly by Cohen himself in purely computational terminology in [14] and depicted in Section 5. *"The immune system functions to compute the state of the body and effects actions accordingly. The input to the immune system is the state of the*

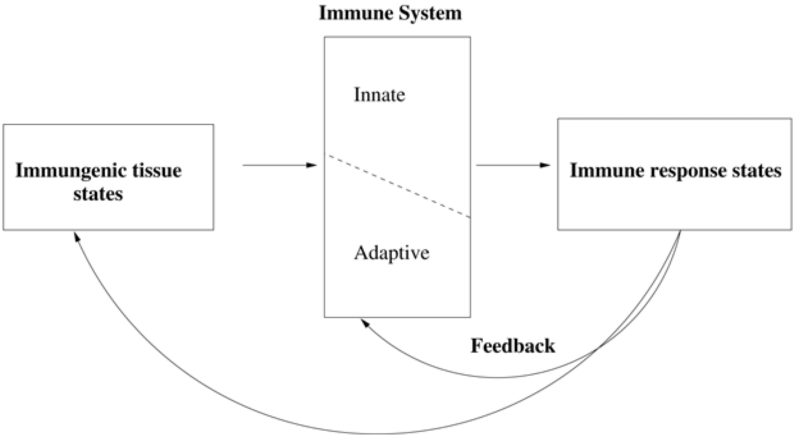


Fig. 5 Immune Computation (adapted from Cohen (2007))

body which comprises of a collection of molecular signals (indicating for example inflammation or trauma). The output is a particular response state which triggers the appropriate processes. The output is fed back to the tissues (inducing healing for example), but also to the immune system itself which modifies both its structure and behaviour. The response is thus formulated as a result of the cumulative experience of the immune system dealing with both the body and the world.”.

In short, the cognitive view is indeed compelling as a framework for building computational systems — one obvious application is in developing autonomic or pervasive systems, which can continuously compute their state and act accordingly. Although Varela [51] and Stewart [45, 46] amongst others have attempted to develop models which capture aspects of this cognitive view, there is no all-encompassing model of the cognitive immune system as a whole (from an immunological perspective anyway). However, some progress has recently been made in formalising a computational framework which captures the ingredients and collaborations of the concept by Voigt *et al* in [54] who propose a modified Learning Classifier System (LCS) approach. The LCS was first introduced by Holland in [30] and is a classic example of a truly collaborative learning system. Parallels between the LCS and the immune system were first drawn by Farmer in 1986 [20], and since then a number of authors have used the LCS framework to model the functioning of the immune system. The relationship between the two systems is discussed in [21] and Vargas *et al* use a simple LCS-immune model in an robotic autonomous navigation application [53, 52]. Voigt *et al* go further however in capturing the characteristic aspects of Cohen’s cognitive view, that of degeneracy and context, in an LCS framework. They point out that the framework offers much potential in the domain of machine-learning and problem-solving, in particular in the context-based processing of signals. In Section 5, we describe an alternative implementation of Cohen’s ideas in research in the wireless sensor network area.

4 Interpreting Immune Collaborations in Real-World Applications

The preceding discussion has emphasised a number of collaborative interactions in the natural immune system which we suggest might be exploited in computational systems in the future. We now show how some of these mechanisms might be interpreted in real-world applications. In the first example, we map open problems in machine learning to immune-inspired mechanisms. This represents work currently in progress. In a second example, some practical work that has already been undertaken in the field of wireless sensor networks is presented. In both cases, the applications possess the properties described in Section 1.2 which suggests they can benefit from an immune-inspired approach. In both cases, the collaborative mechanisms inspired from the immune system provide a means to achieving the desired system functionalities.

4.1 Application of Carneiro's Model in a Machine Learning Scenario

In Section 3 we described Carneiro's model of the immune response which is centered around the interaction of B and T cells and their ability to bind protein. By abstracting these mechanisms, we now show how it is possible to interpret its behaviour in terms of an online learning task. We discuss how we might apply these behaviours to *appropriate* machine learning problems, where appropriateness is defined in terms of similarity to the biological problem, in accordance with the principles described in [28]: life-long online learning of changing concepts of self and non-self based on endogenous criteria. *Online*, because the immune system has no separate training phase or priming via parental genes; *Changing*, because the self is not constant (puberty, pregnancy, transplants etc.); *Endogenous*, because there is no external oracle *a priori* labelling molecules as self or nonself (though there may be environmental feedback). The conceptual motivations that may subsume alternate approaches are discussed.

The immune system functions despite a number of constraints. The first is that it only *senses* molecular shapes within a certain size – those able to bind receptors. Larger compound structures such as protein, complexes, tissues and organisms are invisible, or rather implicit, to the immune response. The second constraint is that chains of amino acids (called "peptides") capable of acting as ligands may often be equally likely to appear in the context of both normal physiology and pathology – the self and non-self are made of the same basic components. The first constraint implies that the task faced by the immune system may be better understood as that of *feature-selection*: distinguishing sufficiently discriminatory features (peptides) from random and persistent noise. The second constraint emphasises that features are not independent labels of selfhood or non-selfhood, but must be judged in context and in relation to the morphological and dynamic dependencies between them. These are critical points; *most* AIS literature [6], relies on either pre-labelling data into one class or another, or presenting examples of one class. The latter approach makes an assumption that data not represented by examples in the presented class belongs to the other which is clearly not valid in a number of applications. Furthermore, the notion that many features of a set of data may not be sufficiently discriminatory to provide any classification ability is generally ignored in AIS literature — all features are implicitly assumed to be discriminatory.

This has a direct mapping to a machine learning context. For example, consider the task of trying to discern relevant information that may be contained in a continuous stream of incoming documents. Words in the documents map to proteins; some of these words can be considered random noise, and therefore should be tolerated. Other words, although not noise, are too ubiquitous to be useful. These words also must be tolerated - the remainder provoke some level of response. Just as in the immune system, where the morphological and dynamic dependencies between proteins determine their current 'label' as opposed to some pre-assigned labelling, the categorisation of a word in a document stream depends not on the word itself but the context it finds itself in. The key factor is that the model does *not* at any stage treat words independently[7], but that the response to any word depends on its interaction with other words. Interactions might include the context the word is found in (e.g. the sentence the word appears in), how similar the word is to other words being presented, and the dynamics of the presentation of the word over time. Determining what to react to and what to tolerate is precisely what the model of tolerance derived from the B-T cell interaction model attempts to achieve. The *co-operation* between T-Clones and B-Clones is the essential mechanism that makes this work:

- The degree of suppression toward T-Clones and their cognate antigen are essentially a judgement on the signal-to-noise ratio of that particular antigenic shape: weakly significant (low-dose) and weakly discriminatory (persistent/high-dose) antigen are actively tolerated, while those in-between invoke various degrees of immune response. The asymmetry and magnitude of the response acts as a confidence margin of the appropriateness of the response.
- The B-Clone repertoire is a constructive representation of the antigenic environment, made viable by clonal selection against present surface patterns. Competitive exclusion over available T-Help regulates the complexity of this representation. The extremes of tolerance and immunity by extinction further emphasises the structure underlying the viable repertoire, by removing inappropriate responding cells and deactivating useful but currently unnecessary cells. The high turnover and disposability of individual components of the repertoire allows the representation to *adapt* rapidly to changes in the environment.

Individual cells in the repertoire have a small window of contribution and contributions overlap considerably. The capacity for successful clones to proliferate lends them extra weight when integrating these responses into an executive *decision*. This

[6] An exception being the DCA algorithm of Greensmith *et al* [25].

[7] Which would simply reduce the model to a statistical model.

increases confidence in the system wide response, even if there is much contradiction at the component level: the variance of individual responses is reduced. Conversely, the higher levels of suppression toward more confident responses changes the antigenic environment of the host. This makes redundant aspects of the repertoire less viable; that is, bias in the preceding responses is compensated in later responses. Furthermore, this additional diversity in the *data* (antigen), as well as diversity in the repertoire, allows the system as a whole to perform *better* than its single best component, which is the theoretical upper bound of an majority vote (see e.g. [33]).

4.2 The Relationship of the Carneiro Model with Theoretical Machine Learning

These ideas share a formal foundation in computational and statistical learning, notably culminating in Boosting by Schapire *et al.* [23, 24]. Boosting is known for its formal demonstration of the “strength of weak learnability” [41]: individual arbitrary weak learners, each performing only marginally better than random guessing, can be aggregated into an arbitrarily strong learning algorithm. This result relies on the diversity generated in the data and the individual learners, essentially as described above. The correspondence, both conceptually and practically, with the immune system is both natural and compelling (see Table 1).

Of course, the immune system is not simply a learning algorithm or statistical model. The immune system does not train, test on independent data, then terminate when its improvements are statistically significant. It cannot make *a priori* assumptions that constrain its learning problem to something analytically convenient. It cannot run several times and choose the best, or average, performance. However, studying the mechanisms of an immune response potentially offers a means by

Table 1 Comparison of conceptual ideas in Boosting and Immune System

Boosting	**Immune System**
Weighted majority to increase confidence and reduce variance	System-wide coherent responses via inter-cell signalling
Reweighting data to increase accuracy and reduce bias	Feedback between antigen presentation, cell proliferation, antibody production and antigen opsonisation
Weak Learner	B-Clone with limited receptor specificity
Strong Learner	Immune repertoire
Regularisation to manage representation complexity	Competitive exclusion over available T-Help to focus the repertoire
Learning algorithm e.g. $y \in \{+1, -1\}$	Competing complementary responses $y = -1$ and $y = +1$

which models might be derived which capture features which are currently outwith the scope of isolated models of statistical learning – how to adapt over time, how to self-regulate, how to co-evolve with adversaries, how to distribute responsibility and manage resources under physical constraints.

The immune system must maintain a healthy host and deploy lethal effector mechanisms. The cost of errors in either case can be fatal or disabling. It runs constantly over the lifetime of the host, interacting with the host's physiology, the external environment and other hosts. This is an application area severely underdeveloped in the computational intelligence literature. It is not difficult to imagine scenarios where these behaviours are essential, particularly as computing devices becomes increasingly ubiquitous, ad-hoc and unmanaged. The immune metaphor provides a coherent, economic and comprehensive framework for thinking about and tackling these domains.

5 A Practical Perspective: Application of Innate and Adaptive Immune Mechanisms to WSN

The previous section described a potential application of Carneiro's model of adaptive immunology to machine learning problems. We conclude the discussion with a look at another challenging domain in which some progress has recently been made in exploiting some of the immune mechanisms described earlier. We describe work in progress in a branch of the wireless sensor network field known as Speckled Computing, which exploits both the Cohen view of the immune system and some of the more specific processes observed in dendritic cells. As in the previous example, the domain possesses exactly the qualities defined by [28] which suggest an immune inspired approach might be useful. The description also provides a direct illustration of how the immune mechanisms can be interpreted in a engineering context. The discussion below presents a high-level overview of concepts involved - the reader is referred to [18] for the actual technical details of the system implementation.

Speckled Computing developed from the field of wireless sensor networks, and sets out to offer a new concept in the way digital and physical worlds communicate and interact [6]. Currently under ongoing development by the SpeckNet consortium, a *speck*, illustrated in Figures 6 and 7, is a miniature device, capable of performing sensing, data processing and wireless networking under program control. The speck is designed to occupy a volume of approximately five millimetres cubed. At such a small scale, energy storage and available memory are highly restricted. Although efficient battery technologies and other forms of renewable power sources are being investigated, operations in specks need to be modest both in terms of power consumption and memory requirements. Moreover, these units are aimed to be inexpensive, produced and used in thousands, suggesting unreliability of specks is an expected feature. A number of key differences between a SpeckNet and a traditional sensor network are apparent [58]. Firstly, SpeckNets are *program-centric* rather than *data-centric*: a traditional sensor network collects data which is transmitted back to some central hub for processing. In contrast, in a SpeckNet, each unit is

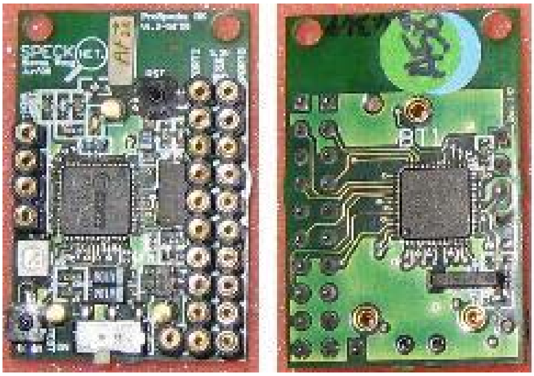


Fig. 6 A version (June 2005) of the speck prototype, ProSpeckz IIK (without battery), built by the SpeckNet Group

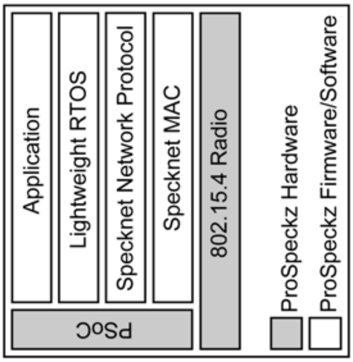


Fig. 7 An overview of the internal architecture of a ProSpeckz [57]

programmable and extracts information and *acts* upon it in collaboration with its local neighbours. Thus, computational tasks as well as data are disseminated through the network. Secondly, specks have a much smaller range of communication than in typical networks, being of the order of tens of centimetres, rather than metres or kilometres. This switches the burden of energy-usage to reception rather than transmission, in contrast to a WSN. Specks are designed to be inherently mobile rather than static, introducing further engineering and software constraints. Finally, SpeckNets operate in an asynchronous manner, with different operations occurring dynamically according to need, with aperiodic data transfer.

Thus, the difficulties associated with the specks' engineering constraints combined with challenging issues in the software development of such a computational system require a fundamentally new approach to development. The interactions between many autonomous, spatially distributed units must result in coherent global behaviours. As powerful central units are not part of the network, the system must find its way to co-ordination through alternative pathways. Furthermore, the system must be able to cope with unpredictable conditions, such as erratic communication and open, sometimes harsh, environments. The domain is a perfect example of a system which exhibits the properties defined by Hart and Timmis in [28] in Section 1.2.

Table 2 Common physiological features between immune system and SpeckNet

Immune System	SpeckNet
Large populations of cells	Great number of specks
Cells distributed in the body	Specks scattered over the environment
Asynchronous molecular signalling	Asynchronous optical/RF communication
Cellular network	Wireless network
Autonomous cells	Self-powered specks
No central organ	No central base unit
Cells simple and expendable	Specks basic and unreliable
Mobile cells	Non-static specks

5.1 *Immune Approaches to SpeckNets*

The vision of the immune system put forward by Cohen as described in Section 3.1 appears to encapsulate at both a structural and physical level the properties that are either observed in a SpeckNet or we wish to emulate. Table 2 outlines the common physiological features between the two system. On a functional level, it is clear that the computational properties expressed in the immune system such as self-regulation, learning, memory and ability and to be robust in the face of unreliable functioning of components and unexpected environmental events, all have direct analogies with a SpeckNet. Above all, both systems comprise of many individually weak components which together must cooperate to produce system wide behaviour.

Although this cognitive view of the immune system is appealing, as already mentioned in Section 3.1, it is difficult to build a computational model which achieves all of the functionality Cohen describes. However, the over-arching themes of the cognitive concepts can still be achieved in a system at a high-level by focusing on modelling some of the components of the natural immune system.

Considering the immune system from a computational point of view, various elements collect information about the state of the body (input), which is channelled through several layers of presentation and filtered collectively by different cells (process), before the appropriate reaction is induced (output). The reaction closes an important feedback loop by causing changes to the body environment and modifying the structure of the immune system itself [14]. Correspondingly, in a SpeckNet, a deployed network must actively gather sensor data which, depending on the defined application, it then must interpret and proceed with an appropriate reaction, if necessary. Throughout its operation, the network is also responsible for attending to itself, i.e managing its internal state (e.g. resources, potential failures) and potentially its structure. The first process in this continuous cycle has a natural mapping to the functioning of dendritic cells, the sentinels of the immune system described in Section 2.3. In particular, the following functional properties of dendritic cells are of interest:

1. Dendritic cells *circulate* through body tissues, sampling exogenous and endogenous signals.
2. Dendritic cells return to the lymph nodes when they become mature, where they deliver a *snapshot of the current environment*.
3. The maturation state under which DCs return to the lymph provides crucial information to the system regarding *how* it should react.
4. The lymph nodes in the body are *distributed;* the large lymph nodes are strategically located to areas of the body that are closer to sources of input from the environment.

From this in [16, 18] we have derived the outline of a model which maps tissues in the body to specks, and messages sent between specks to dendritic cells. We currently distinguish two different types of specks:

- *Tissue* specks correlate to tissues in the body, and contain sensors for monitoring the external environment (e.g. pressure, temperature etc.). They can also provide endogenous signals, for example relating to their own internal state (i.e. battery power). These specks constitute the majority of specks in any given environment.
- *Integration* specks correspond to lymph nodes. These specks receive information from dendritic cells, process it, and determine an appropriate response. These specks *may* have greater processing power than tissue specks (but are not required to).

A typical environment will contain a high ratio of tissue specks over integration specks. Although in the body, lymph nodes are strategically placed, this is not feasible in a typical speck deployment, where ultimately, applications are envisaged in which thousands of specks may be sprayed at random into an environment. To take account of this, we have studied a number of models of SpeckNets which adopt random placements of integration specks. Dendritic cells are mapped to *scouting messages*. Messages originate at integration specks and traverse the tissue specks, where they collect both exogenous and endogenous environmental information from each speck visited. Eventually they return to an integration speck where the information collected is filtered and aggregated. Eventually, a decision may be made by the integration node to act upon the collective information. This may result in one or more of several possible actions. For example, consider an application in which a SpeckNet might be used to monitor *temperature* in an environment. As well as maintaining a functioning network, the SpeckNet should be able to regulate fluctuations in temperature in local regions by activating or deactivating heat sources, as well as indicate unusual sources of heat, for example, a fire. Integration nodes therefore might send out effector messages which modify the external environment, e.g turning a heat source on or off; alternatively, the integration node may modify the endogenous variables of the system, for example, alerting tissue specks to modify their endogenous parameters or increasing the rate at which it sends out scouting messages in order to gather more information.

An overview of the model is given in Figure 8. Scouting messages originate from an integration speck and are relayed through the SpeckNet in essentially a random

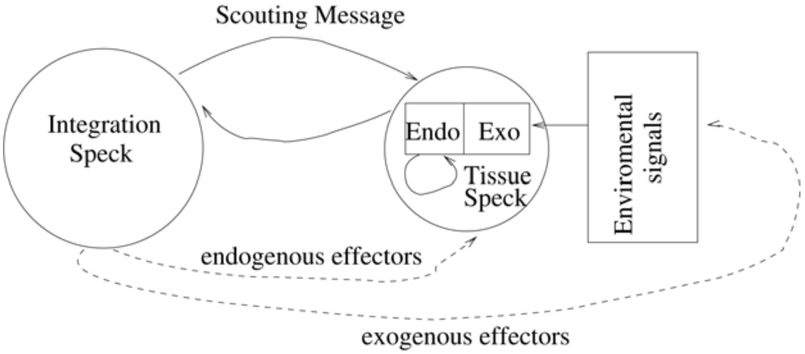


Fig. 8 A high level overview of immune-inspired model used to achieve achieve functionality and self-regulation and maintenance in a SpeckNet

walk. As they visit tissue specks, they collect information from the speck regarding its local perception of the environment. For example, this may take the form of an average value of a sensor reading over a specified window size. The parameters used for processing sensor readings from the tissues and the functions applied to local sensor values depend on the application and are set accordingly. The scouting message aggregates this information as it travels (again in an application dependent manner) and eventually either expires, matures or semi-matures depending on the information it has collected. The conditions which cause a scouting message to mature may for example include measuring variance of exogenous signals or monitoring of endogenous signals such as battery power. This change of state triggers a return to the integration node where the scouting message presents its information. The integration node is able to estimate context based on the proportion of different types of messages returning and also by aggregating information contained in the expired messages. Once again, aggregating information from the network via scouting messages is configurable with respect to the type of information needed.

The model just described has been tested in simulation using a simulation tool, SpeckSim [1], developed to simulate a SpeckNet environment, taking account of the physical constraint apparent in such a network. For example, radio communication is asynchronous, messages can be lost, and specks can go down at any time (either temporarily or permanently). Experimentation with SpeckNets under a number of random configurations using a heat-monitoring scenario has confirmed that integration nodes can successfully obtain a local picture of their environment, and detect local changes. There is clearly a balance to be struck concerning the locality of the information obtained by an integration node — increasing the number of these nodes in the system results in each node obtaining a very local picture; fewer nodes give a more global overview. [16, 18] describe these experiments in detail, presenting technical results. Continuing research in this vein is now focusing on the processes that occur in the integration nodes which integrate the information from returning scouting messages and formulate a response. One avenue of investigation focuses on collaborative voting methods, where a majority vote based on returned states might determine the eventual outcome. Another research direction will focus on the collaboration between B and T-cells evident in the Carneiro model described in Section 3 which allow the integration nodes to *learn* a model which will enable tolerance or reactivity to naturally emerge in a system based on the stream of information arriving at the nodes.

6 The Future of AIS: Immuno-Engineering

The previous section has illustrated how one example of a collaborative mechanism in the natural immune system can be practically exploited in an engineered system. Taking a step back from this, we conclude the paper by examining how systems such as the one described might be designed in the future via a principled approach which results in generic, well-understood mechanisms which have wide applicability to a range of domains, rather than being problem or domain specific.

Timmis *et al* propose a new branch of engineering to be known as *immuno-engineering* which is inspired by the work of Orosz [39]. Orosz defines *immuno-ecology* and *immuno-informatics* (definitions 1 and 2 respectively), stating that *immuno-informatics addresses the mechanisms by which the immune system converts stimuli into information, how it processes and communicates that information, and how the information is used to promote an effective immuno-ecology. [39].*

Definition 1. *Immuno-ecology is the study of immunological principles that permit effective immunological function within the context of the immensely complex immunological network*

Definition 2. *Immuno-informatics is the study of the immune system as a cognitive, decision-making device*

Following on from this, Timmis *et al* outline a vision for a new type of engineering they term *immuno-engineering* (definition 3) in [49], which they argue can be used for the development of biologically grounded and theoretically understood AIS. This is envisaged to enable the construction of robust, engineered artifacts via a bottom-up approach to engineering.

Definition 3. *Immuno-engineering is the abstraction of immuno-ecological and immuno-informatics principles, and their adaptation and application to engineered artifacts (comprising hardware and software), so as to provide these artifacts with properties analogous to those provided to organisms by their natural immune systems.*

Immuno-engineering takes into account the differences between artificial systems and biological systems: for example, the different numbers, kinds, and rates of signals that need to be monitored and processed; the different kinds of decisions that need to be made; the different effectors available to support and implement those decisions and the different constraints of embodiment. For example, Orosz [39] suggests that the most important *design* feature of the immune system that endows it with speed, flexibility and multiple-response options is the parallel-processing

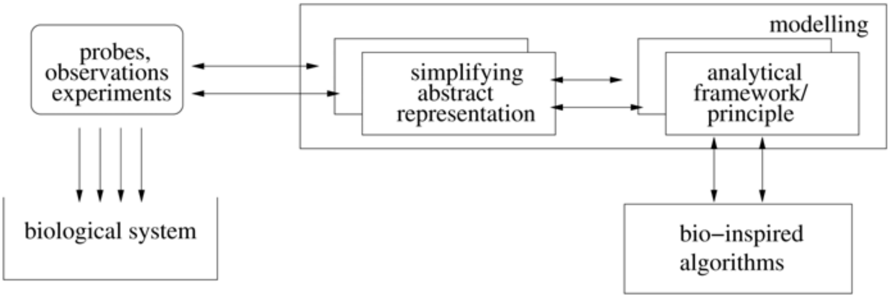

modelling
probes, observations experiments
simplifying abstract representation
analytical framework/ principle
biological system
bio–inspired algorithms

Fig. 9 The conceptual framework proposed by Stepney *et al*, taken from [44]

architecture of the system. However, such an architecture is extremely wasteful in its use of resources and relies on numerous back-up systems. The natural system cannot simply be translated to an engineered system - the engineer is constrained by physical constraints such as processing speeds, communication overheads and physical resource and further hindered by hardware requirements such as transmitting signals from devices. [49] argue that in order to successfully transfer the *functionality* of the immune system to an engineered system, one must follow a principled approach, probing the biological system and building computational and mathematical models which can be executed and validated. To this end, they advocate the use of the *conceptual framework* originally proposed by Stepney *et al* which provides a structure which allows principled models to be derived which capture the rich functionalities of the underlying systems. The framework is shown in Figure 9. The framework advocates that probes, for example, observations and experiments, be used to obtain a view of a complex biological system. Although any such view is clearly limited to a sub-part of the complete system, it can be used to build abstraction models which attempt to capture the underlying biology. These models can then be validated. The models, which may be mathematical or computational in nature, can then be further abstracted in analytical computational frameworks which can also be validated. Depending on the desired application, the validation may require use of benchmark problems or engineering demonstrators. Above all, the framework, in line with [48] emphasises the need for collaborative, interdisciplinary research to sophisticated models built on well-founded analytical principles.

The ultimate goal of the research presented in [49] is to be able to use such a framework to create a generic set of immuno-engineering libraries which encapsulate essential properties and interactions and can be instantiated in a variety of applications. It is hoped that in the future this will push the boundaries of both biologically inspired computing and engineering.

7 Conclusions

The natural immune system is a vastly complex system, which functions due to a complicated web of interactions that occur between multiple cells and multiple signals. Although AIS has developed a strong, and thriving field in its own right over the past decade or so, in this article we have argued that the true potential of the immune system has not yet been exploited in computational systems. The metaphor is much richer than one might perceive given a glance through the AIS literature. We have outlined some of the functional properties of the immune system that emerge as a result of these collaborations, and shown how these properties are desirable, if not essential, in engineered systems, particularly those that are required to function autonomously. Some of the immune mechanisms which might be exploited to achieve this functionality have been described in order to point the reader in the right direction. These mechanisms must be considered carefully in order to prevent a recourse to reasoning by metaphor - the conceptual framework and the immuno-engineering approach described provided a scientifically appropriate method for achieving this. Any field, whatever the inspiration, must ultimately prove its worth by standing on firm theoretical foundations, rather than simply exploiting its novelty. Exploiting the similarities between immune mechanisms and other more traditional domains such the boosting domain discussed earlier in the article may offer some mileage in this respect. Progress may also be made via the *immuno-engineering* approach described in Section 6 which advocates a principled abstraction of immune metaphors, through a series of abstractions, at a mathematical and computational level.

Our argument is perhaps summed up most succinctly by the words of Neal *et al* in [37] who propose

> ...the importance of the nature of the interactions in the immune system leads naturally to the expectation that far more complex and ambitious immune inspired computation than is currently attempted is required and should be possible.

By considering complex applications, we look forward to a new age in AIS in which it may be possible to construct engineered artifacts that are fit for purpose in the same way as their biological counterparts.

References

1. `http://www.specknet.org/dev/specksim`
2. Artificial Immune Systems, Proceedings of International Conferences, Lecture Notes in Computer Science. Springer (2001-2008)
3. Andrews, P., Timmis, J. In: Silico Immunology, chap. Alternative Inspiration for Artificial Immune Systems: Exploiting Cohen's Cognitive Immune Model. Springer (2007)
4. Andrews, P., Timmis, J. In: Silico Immunology, chap. Alternative Inspiration for Artificial Immune Systems: Exploiting Cohen's Cognitive Model. Springer (2007)
5. Andrews, P.S., Timmis, J.: Inspiration for the next generation of artificial immune systems. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005, vol. 3627, pp. 126–138. Springer, Heidelberg (2005)
6. Arvind, D., Elgaid, K., Krauss, T., Paterson, A., Stewart, R., Thayne, I.: Towards an integrated design approach to specknets. In: IEEE Int. Conf. Communications 2007, ICC 2007, pp. 3319–3324 (2007)

7. Bersini, H.: Why the first glass of wine is better than the seventh. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 100–111. Springer, Heidelberg (2005)
8. Burnet, F.: The clonal selection theory of acquired immunity. Cambridge University Press, Cambridge (1959)
9. Carneiro, J., Coutinho, A., Faro, J., Stewart, J.: A model of the immune network with b-t cell co-operation. i - prototypical structures and dynamics. Journal of Theoretical Biology 182, 513–529 (1996)
10. Carneiro, J., Coutinho, A., Stewart, J.: A model of the immune network with b-t cell co-operation. ii - the simulation of ontogenisis. Journal of Theoretical Biology 182, 531–547 (1996)
11. de Castro, L., Von Zuben, F.: Learning and optimization using the clonal selection principle. IEEE Transactions on Evolutionary Computation 6(3), 239–251 (2002)
12. Cohen, I.: The cognitive paradigm and the immunological homunculus. Immunology Today (1992)
13. Cohen, I.: Tending Adam's garden: evolving the cognitive immune self. Elsevier Academic Press, Amsterdam (2000)
14. Cohen, I.: Real and artifical immune systems: computing the state of the body. Nature (2007)
15. Coutinho, A.: Immunology at the crossroads. European Molecular Biology Organisation Reports 3(11), 1008–1011 (2002)
16. Davoudani, D., Hart, E.: Computing the state of specknets: An immune-inspired approach. In: To be published in Proc. Int. Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS 2008, Edinburgh, UK, June 16–18 (2008)
17. Davoudani, D., Hart, E., Paechter, B.: An immune-inspired approach to speckled computing. In: de Castro, L.N., Von Zuben, F.J., Knidel, H. (eds.) ICARIS 2007. LNCS, vol. 4628, pp. 288–299. Springer, Heidelberg (2007)
18. Davoudani, D., Hart, E., Paechter, B.: Computing the state of specknets: Further analysis of an innate immune-inspired model. In: Bentley, P.J., Lee, D., Jung, S. (eds.) ICARIS 2008. LNCS, vol. 5132, pp. 95–106. Springer, Heidelberg (2008)
19. De Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, London (2002)
20. Farmer, J.D., Packard, N.H., Perelson, A.S.: The immune system, adaptation and machine learning. Physica 22, 187–204 (1986)
21. Forrest, S., Hofmeyr, S., Somayaji, A.: Computer immunology. Commun. ACM 40(10), 88–96 (1997)
22. Forrest, S., Perelson, A., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: Proceedings of the IEEE Symposium on research, security and privacy, pp. 202–212 (1994)
23. Freund, Y., Schapire, R.E.: A decision theoretic generalisation of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
24. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting (1998), http://citeseer.ist.psu.edu/friedman98additive.html
25. Greensmith, J., Aickelin, U., Twycross, J.: Articulation and clarification of the dendritic cell algorithm. In: Bersini, H., Carneiro, J. (eds.) ICARIS 2006. LNCS, vol. 4163, pp. 404–417. Springer, Heidelberg (2006)

26. Hart, E., Bersini, H., Santos, F.: How affinity influences tolerance in an idiotypic network. J. Theor. Biology (2007)
27. Hart, E., Davoudani, D., McEwan, C.: Immunological inspiration for building a new generation of autonomic systems. In: Autonomics, p. 9 (2007)
28. Hart, E., Timmis, J.: Application areas of ais: The past, the present and the future. Applied Soft Computing 8, 191–201 (2008)
29. Hofmeyr, S., Forrest, S.: Immunity by design. In: Proceedings of GECCO 1999, pp. 1289–1296 (1999)
30. Holland, J.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge (1992)
31. Janeway, C.A., Travers, P., Walport, M., Schlomchik, M.: Immunobiology. Garland (2001)
32. Jerne, N.: Towards a network theory of the immune system. Annals of Immunology (Inst. Pasteur) 125, 373–389 (1974)
33. Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. Information and Computation 108, 212–261 (1994)
34. McEwan, C., Hart, E., Paechter, B.: Revisiting the central and peripheral immune system. In: de Castro, L.N., Von Zuben, F.J., Knidel, H. (eds.) ICARIS 2007. LNCS, vol. 4628, pp. 240–251. Springer, Heidelberg (2007)
35. McEwan, C., Hart, E., Paechter, B.: Towards a model of immunological tolerance and autonomous learning. Submitted to Natural Computing (2008)
36. Neal, M.: Meta-stable memory in an artificial immune network. In: Timmis, J., Bentley, P.J., Hart, E. (eds.) ICARIS 2003. LNCS, vol. 2787, pp. 168–180. Springer, Heidelberg (2003)
37. Neal, M., Trapnell, B.J.: Go Dutch: Exploit Interactions and Environments with Artificial Immune Systems. In: Silico Immunology, Springer, Heidelberg (2007)
38. Neuman, Y.: The immune self, the sign and the testes. Semiotics, Evolution, Energy, Development, 85–109 (2005)
39. Orosz, C.: Desgin Principles for Immune System and Other Distributed Autonomous Systems, chap. An Introduction to Immuno-ecology and Immuno-informatics. Oxforf Univ.Press, Oxforf (2001)
40. Perelson, A.S., Weisbuch, G.: Immunology for physicists. Review of Modern Physics 69 (1997)
41. Schapire, R.E.: The strength of weak learnability. Machine Learning 5, 197–227 (1990), http://citeseer.ist.psu.edu/schapire90strength.html
42. Sompayrac, L.: How the immune system works, 3rd edn. Blackwell Publishing, Malden (2008)
43. Stepney, S.: In Silico Immunology, chap. Embodiment. Springer, Heidelberg (2006)
44. Stepney, S., Smith, R., Timmis, J., Tyrrell, A., Neal, M., Hone, A.: Conceptual frameworks for artificial immune systems. Journal of Unconventional Computing 1(3), 315–338 (2005)
45. Stewart, J.: Cognition without neurons: adaptation, learning and memory in the immune system. In: CC AI, pp. 7–30 (1994)
46. Stewart, J., Coutinho, A.: The affirmation of self: A new perspective on the immune system. Artificial Life 10, 261–276 (2004)
47. Stibor, T., Timmis, J., Eckert, C.: On the use of hyperspheres in artificial immune systems as antibody recognition regions. In: Bersini, H., Carneiro, J. (eds.) ICARIS 2006. LNCS, vol. 4163, pp. 215–228. Springer, Heidelberg (2006)
48. Timmis, J., Andrews, P., Owens, N., Clark, E.: An interdisciplinary perspective on artificial immune systems. Evolutionary Intelligence 1(1), 5–26 (2008)

49. Timmis, J., Hart, E., Neal, M., Stepney, S., Tyrrell, A.: Immuno-engineering. In: 2nd IFIP International Conference on Biologically Inspired Collaborative Computing. IEEE Press, Los Alamitos (2008)
50. Timmmis, J.: Artificial immune systems - today and tomorrow. Natural Computing 6(1), 1–18 (2007)
51. Varela, F., Coutinho, A., Dupire, B., Vaz, n.: Cognitive networks: Immune, neural and otherwise. J. Theoretical Immunology (1988)
52. Vargas, P., de Castro, L., Michelan, R., Von Zuben, F.: An immune learning classifier network for autonomous navigation. In: Timmis, J., Bentley, P.J., Hart, E. (eds.) ICARIS 2003. LNCS, vol. 2787, pp. 69–80. Springer, Heidelberg (2003)
53. Vargas, P., de Castro, L., Von Zuben, F.: Mapping artificial immune systems into learning classifier systems. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 2003. LNCS (LNAI), vol. 2661, pp. 163–186. Springer, Heidelberg (2003)
54. Voigt, D., Wirth, H., Dilger, W.: A computational model for the cognitive immune system theory based on learning classifier systems. In: de Castro, L.N., Von Zuben, F.J., Knidel, H. (eds.) ICARIS 2007. LNCS, vol. 4628, pp. 264–275. Springer, Heidelberg (2007)
55. Watkins, A., Timmis, J.: Exploiting parallelism inherent in airs: an artificial immune classifier. In: Nicosia, G., Cutello, V., Bentley, P.J., Timmis, J. (eds.) ICARIS 2004. LNCS, vol. 3239, pp. 427–438. Springer, Heidelberg (2004)
56. Whitbrook, A., Aickelin, U.J.G.: Idiotypic immune networks in mobile robot control. IEEE Transactions on Systems, Man and Cybernetics, Part B 37(6), 1581–1598 (2007)
57. Wong, K., Arvind, D.: Speckled computing: Disruptive technology for networked information appliances. In: Proceedings of the IEEE International Symposium on Consumer Electronics (ISCE 2004), pp. 219–223 (2004)
58. Wong, K., Arvind, D.K.: Specknets: New challenges for wireless communication protocols. In: Third International Conference on Information Technology and Applications. ICITA 2005, vol. 2, pp. 728–733 (2005)
59. Zambonelli, F., Van Dyke Parunak, H.: Signs of a revolution in computer science and software engineering. In: Petta, P., Tolksdorf, R., Zambonelli, F. (eds.) ESAW 2002. LNCS, vol. 2577, pp. 13–28. Springer, Heidelberg (2003) (revised papers)

Part VIII
Parallel Evolutionary Algorithms

Evolutionary Computation: Centralized, Parallel or Collaborative

Heinz Mühlenbein

Abstract. This chapter discusses the nature and the importance of spatial interactions in evolutionary computation. The current state of evolution theories is discussed. An interaction model is investigated which we have called Darwin's *continent-island cycle* conjecture. Darwin argued that such a cycle is the most efficient for successful evolution. This bold conjecture has not yet been noticed in science. We confirm Darwin's conjecture using an evolutionary game based on the iterated prisoner's dilemma. A different interaction scheme, called the stepping-stone model is used by the Parallel Genetic Algorithm PGA. The PGA is used to solve combinatorial optimization problems. Then the Breeder Genetic Algorithm BGA used for global optimization of continuous functions is described. The BGA uses competition between subpopulations applying different strategies. This kind of interaction is found in ecological systems.

1 Introduction

Modeling evolutionary principles found in nature in order to make the development of powerful problem solving programs easier, or even to create human-like intelligence was tried already in the beginning of the computer area. At least three approaches have been followed to achieve this goal.

1. **Use a theory** - develop a theory of problem solving and implement it on a computer
2. **Copy the brain** - analyze the human brain and make a copy of it on a computer
3. **Copy natural evolution** - analyze natural evolution and implement the most important evolutionary forces on a computer

In the history of artificial intelligence research one of the three approaches was dominant at any one time. The second and third approach depend on biological

Heinz Mühlenbein
Fraunhofer Institut Autonomous intelligent Systems Schloss Birlinghoven 53757 Sankt Augustin, Germany
e-mail: heinz.muehlenbein@online.de

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 561–595.
springerlink.com

knowledge. In this chapter I will concentrate on the third approach. It relies on theories of evolution and of computation. The theory of computation is well advanced, so the problems of evolutionary computation lie in implementing theories of evolution. If a convincing constructive (or even mathematical) theory of evolution existed, then evolutionary computation would be just a matter of implementation - which of the major evolutionary forces to implement in what detail.

But does biology possess a constructive theory of evolution? Here the opinions differ considerably. The main stream theory of evolution is called *New* or *Modern Synthesis*. Its followers claim that it reconciles Darwin's idea of continuous small variations with gene flows derived from population genetics. The second major force of the Modern Synthesis is Darwin's concept of *natural selection*. But are these two forces sufficient to explain the wonders of evolution at least in some broad terms?

There is no doubt that Modern Synthesis is able to explain the change of gene frequencies on a small time scale. If there is enough diversification, then the theory correctly predicts further changes for a short time. But can it explain the evolution on a large time scale with new species arising and old species vanishing?

The outline of the chapter is as follows. First I recall the state of art of evolution theories, because they are used as models for evolutionary computation. Then I describe different genetic algorithms, centralized, parallel and co-evolutionary. The major part of the chapter deals with the investigation of Darwin's conviction that space is as important as time for evolution to take place. Especially we[1] will analyze an important conjecture of Darwin which has been unnoticed sofar. We have called it the *Continent-island cycle conjecture*. This conjecture is analyzed using evolutionary games. Here a number of different spatial distributions are compared. Then I describe the parallel genetic algorithm PGA and its use in combinatorial optimization. In the final section co-evolution of sub-populations is used for continuous optimization.

The term collaboration does not appear in textbooks of biology. In a restricted form collaboration is investigated in ecology. Collaboration in the general sense is considered to be an important component of human societies and therefore part of sociology. Biology researches cooperation driven by interactions - between individuals, between species, between geographic distributed sub-populations, within insect colonies etc. In this chapter we investigate spatial distributions which vary over time. The most interesting distribution is the continent-island cycle. This might also be a model for successful collaboration in human societies.

2 Darwinism - The Unfinished Theory

Darwin‘s book “The Origin of Species by Means of Natural Selection” had undoubtedly an immense influence on biology and modern thinking in general. But it should have been seen as the beginning of a new theory of evolution, not a solution. The further development of Darwin‘s ideas has been hindered especially

[1] The research has been done together with many researchers. Therefore I use mainly we throughout the chapter.

in the Anglo-Saxon countries because of the battle of its supporters against some orthodox believers of religion[2].

In Germany Ernst Haeckel was a strong advocate of Darwin's theory. Nevertheless he wrote as early as 1863 - only four years after the publication of the Origin: "Darwin's evolution theory is by no means finished, instead it gives only the first outline of a future theory. On the one hand we are not aware of all the other relations, which may be equally important in the origin of species to natural selection, which was emphasized far too much by Darwin. And in many cases the external conditions of existence of an-organic nature like climate and habitat, geographic and topographic conditions, to which the organism have to adapt, should be considered no less important than these relations....Another, and no doubt the most important shortcoming of Darwin's theory lies in the fact, that it gives *no indication of the spontaneous creation or the first genesis of the one or the few oldest organisms from which all other organisms developed [19]*[3]".

It is outside the scope of this paper to discuss the above problems in detail. They are still controversial in biology. In order to refresh the memory of the reader I recall some important terms in evolutionary biology

- **Genotype:** The molecular basis of inheritance as defined by genes and chromosomes.
- **Phenotype:** The actual appearance of the living beings.
- **Species:** A group of organisms capable of inter-breeding and producing fertile offspring. More precise measures are based on the similarity of DNA or morphology.

Another important concept in population genetics is the *fitness*. It describes the capability of an individual of certain genotype to reproduce, and usually is equal to the proportion of individual genes in the *next* generation. An individual's fitness is manifested through its phenotype. As the phenotype is affected by both genes *and* environment, the fitness of different individuals with the same genotype are not necessarily equal, but depend on the environment on which the individuals live. However, the fitness of the genotype is considered to be an averaged quantity, it will reflect the outcomes of all individuals with that genotype.

This is a very careful definition, but how can this fitness be measured? It needs the next generation! I will not discuss this complicated issue further. All mathematical models of population genetics assume that the fitness is given. In the simplest case of a single gene with two alleles a and A, we have the genotypes with genotypes aa, aA, AA with corresponding fitness values w_{00}, w_{01}, w_{11}.

In order to illustrate the current state of art of evolution theories, I shortly describe two representative examples. The first one is expressed in the book of Maynard Smith and Szathmary [47]. They see evolution as the evolution of complexity in terms of genetic information and how it is stored, transmitted, and translated. This

[2] This controversy is still not settled, if one considers the many supporters of "intelligent design" in the US.

[3] Translation by the author.

Table 1 Major transitions in evolution [47]

before	→ after
replicator molecules	→ population of molecules in compartments
independent replicator	→ chromosomes
RNA as gene and enzyme	→ DNA and protein
procaryote	→ eucaryote
asexual clones	→ sexual population
protist	→ plants, animals, fungi
solitary individuals	→ colonies
societies of primates	→ human societies

approach has led them to identify several *major transitions*, starting with the origin of life and ending with the origin of human language (see Table 1).

The authors "solve'" some of the transition problems with a very narrow version of the Modern Synthesis. "We are supporters of the gene centered approach proposed by Williams and refined by Dawkins." In the gene centered approach, also called the *selfish gene concept* [7], the genes are the major actors. They possess an internal force to proliferate as much as possible.

Let me illustrate the gene centered approach with the *kin selection* concept. In the gene centered approach fitness measures the quantities of copies of the genes of an individual in the next generation. It doesn't really matter how the genes arrive in the next generation. That is, for an individual it is equal beneficial to reproduce itself, or to help relatives with similar genes to reproduce, as long as at least the same number of copies of the individual's genes get passed on to the next generation. Selection which promotes this kind of helper behavior is called kin selection. It has even been put into a mathematical rule by Hamilton [20]! An altruistic act will be done if

$$C < r * B \tag{1}$$

Here C means the cost in fitness to the actor, B the benefit in fitness and r the relatedness. Let us discuss a simple example. Consider a father and his children, which are drowning. Here $r = 0.5$. Let us assume that the father has to make a sacrifice, this means $C = 1$. First assume $B = 1$. Then the father will not make a sacrifice for a single child, but it needs three children at least. But if the father is not able to father new children, then he makes the sacrifice for a single child! (see also the discussion of altruism in [47]).

The selfish gene concept has been opposed by a small group in biology, most notably the late Stephen J. Gould. Recently even philosophers of science formulate a basic critic. "The synthetic theory bypassed what were at the time intractable questions of the actual relationship between stretches of chromosomes and phenotypic traits. Although it was accepted that genes must, in reality, generate phenotypic differences through interaction with other genes and other factors in development, genes were treated as *black boxes* that could be relied on to produce phenotypic variation with which they were known to correlate [17]."

A more complex theory of evolution is contained in the book by Jablonka and Zeligowski. The title is: "Evolution in Four Dimensions" [23]. The four dimensions are

- The genetic inheritance system
- The epigenetic inheritance system
- The behavioral inheritance system
- The symbolic inheritance system

The genetic dimension dominates the current research. It culminated in sequencing the human genome. The epigenetic inheritance system is less understood and varies considerably. Although their DNA sequences remain unchanged during development, cells nevertheless acquire information that they can pass to their progeny. Let me give a simple example. Liver cells, skin cells, and kidney cells, look different, behave differently, and function differently, yet they contain all the same genetic information. The differences are the consequences of events that occurred during the developmental history of each type of cell and determined which genes are turned on. Now, when liver cells divide their daughter cells are liver cells, and so on. Although their DNA sequences remain unchanged during development, cells nevertheless acquire information that they can pass to their progeny. This information is transmitted by the *epigenetic inheritance system*. Sometimes the information can be transmitted even transgenerationally. The existence of epigenetic inheritance was barely recognized until the mid-1970s.

Behavioral inheritance is still controversial. Most evolutionists stress the genetic basis of behavior (see Maynard Smith discussed earlier). They maintain that behavioral strategies are to a large extent genetically determined and almost independent of each other. Each behavior has been shaped through natural selection of genes that led to the construction of a specific behavioral module in the brain.

The fourth dimension, the symbolic interaction system, is unique to man. There exists only a few attempts of a mathematical theory of the symbolic inheritance system. Most notably is the theory of "cultural transmission" [4].

But what makes even this very condensed theory so difficult is the *interaction of all four inheritance systems*. The genetic inheritance system neither dominates nor even forms a foundation for the other three systems. The four dimensions cannot be analyzed separately, it is their complex interaction which defines the success of the individual.

Metaphorically speaking: Each organism travels on a unique trace in this four dimensional space.

2.1 The System View of Evolution

One of the major weakness of the Modern Synthesis is the *separation of the individuals and the environment*. The fitness is averaged over individuals and environments. Let $\mathbf{O}(\mathbf{t}) = (O_1(t), \ldots O_N(t))$ denote the vector of individuals at generation t. Then we can formulate a simple system model. Each individual $O_i(t)$ (mainly characterized by its genotype) is assigned a fitness f predicting the performance of this individual within the environment E and given the other individuals. Then the evolution can be written as:

$$O_i(t+1) = f(\mathbf{O(t)},E(t)) \tag{2}$$

$$E(t+1) = g(E(t) \tag{3}$$

It seems impossible to obtain numerical values for this fitness. Therefore theoretical biology has made many simplifications. The environment is kept fixed, i.e $g(E(t)) = const$, the influence of other individuals is described by some averages of the population, etc.. The above model is still too simple, because each individual is *developing in a close interaction with its environment*.

The model given by 2 has not yet been used in population genetics, but specialized cases are applied commonly in *population dynamics* [21] or *ecology*. Given two species with population sizes N and M, the following equations are used

$$N(t+1) = F(N(t),M(t)) \tag{4}$$

$$M(t+1) = G(N(t),M(t)) \tag{5}$$

The population sizes of the next generation depends on the interaction of the two species at generation t. The interaction can be positive, this means that both species are supporting each other. If the interaction is negative we have the classical predator-prey system [18].

The development problem in evolutionary models has been addressed recently by the *developmental system theory* [40]. Unfortunately the theory is very informal, it has been formulated from a philosopher's point of view. Therefore I will describe the nucleus of an evolution theory as it has been stated by Anatol Rapaport [43].

The theory is based on the concept of an *organism*. "According to a soft definition, a system is a portion of the world that is perceived as a unit and that is able to maintain its identity in spite of changes going on in it. An example of a system par excellence is a living organism. But a city, a nation, a business firm, a university are organisms of a sort. These systems are too complex to be described in terms of succession of states or by mathematical methods. Nevertheless they can be subjected to methodological investigations [43]."

Rapaport then defines: "Three fundamental properties of an organism appear in all organism-like systems. Each has a structure. That is, it consists of inter-related parts. It maintains a short-term steady state. That is to say, it reacts to changes in the environment in whatever way is required to maintain its integrity. It functions. It undergoes slow, long term changes. It grows, develops, or evolves. Or it degenerates, disintegrates, dies. Organisms, ecological systems, nations, institutions, all have these three attributes: *structure, function, and history*, or, if you will, *being, acting, and becoming*."

Taking the importance of individual development into account, I divide becoming into *developing and evolving*. Development is the process creating a grown-up individual in a given environment from an fertilized egg . Evolution is the change of the species in many generations.

The neuro-biologist Humberto Matura [32] also tried to find criteria for what characterizes life. He proposed a new concept which he called *autopoiesis*. It is derived from Greek "autos=self" and "poiein=to make". The theory remained rather informal and speculative. Therefore the concept was not very successful in biology, but it is very popular in other sciences, especially in sociology.

The major conclusion of this section is: *There exists no theory of evolution today which could serve as a computer model for evolving artificial creatures with problem solving capacities.* Given this situation it is no surprise that early models of evolutionary computation are not based on detailed biological models, but use biological terms only metaphorically. This observation is still valid.

3 Evolutionary Algorithms - Centralized, Parallel or Collaborative

There exist a myriad of evolutionary algorithms which model parts of general evolutionary models described in the previous section. The most popular algorithm is the genetic algorithm GA which models evolution by sexual reproduction and natural selection. The GA was invented by Holland [22]. The optimization problem is given by a fitness function $F(x)$.

Genetic Algorithm

```
1   Define a genetic representation of the problem; set $t = 0$
1   Create an initial population $P(0) = x_1^0, \dots x_N^0$
1   Compute the average fitness $\overline{F} = \sum_i^N F(x_i)/N$. Assign each individual the
    normalized fitness value $F(x_i^t)/\overline{F}$
1   Assign each $x_i$ a probability $p(x_i, t)$ proportional to its normalized fitness.
    Using this distribution, randomly select $N$ vectors from $P(t)$. This gives
    the set $S(t)$
1   Pair all of the vectors in $S(t)$ at random forming $N/2$ pairs. Apply
    crossover with probability $p_{cross}$ to each pair and other genetic operators
    such as mutation, forming a new population $P_{t+1}$
1   Set $t = t + 1$, return to STEP2
```

In the simplest case the genetic representation is just a bit-string of length n, the *chromosome*. The positions of the strings are called *loci* (sing. locus) of the chromosome. The variable at a locus is called a *gene*, its value an *allele*. The set of chromosomes is called the *genotype* which defines a *phenotype* (the individual) with a certain fitness. The *crossover operator* links two searches. Part of the chromosome of one individual (search point) is inserted into the second chromosome giving a new individual (search point). We will later show with examples why and when crossover helps the search.

A genetic algorithm is a *parallel random search with centralized control*. The centralized part is the selection schedule and the mating. For selection the average fitness of the population is needed. The result is a highly synchronized algorithm, which is difficult to implement efficiently on parallel computers.

In our parallel genetic algorithm PGA we use a *distributed selection scheme*. This is achieved as follows. Each individual does the selection by itself. It looks for a partner in its neighborhood only. The set of neighborhoods defines a spatial population structure. Thus the PGA runs *totally asynchronous*, there is no synchronization

Our second major change can now easily be understood. Each individual is active and not acted on. It may improve its fitness during its lifetime by performing a *local search*. The generic PGA can be described as follows

Parallel genetic algorithm

1 Define a genetic representation of the problem
1 Create an initial population and its population structure
1 Each individual does local hill-climbing
1 Each individual selects a partner for mating in its neighborhood
1 An offspring is created with genetic crossover of the parents
1 The offspring does local hill-climbing. It replaces the parent, if it is better than some criterion (acceptance)
1 If not finished, return to STEP3.

Because each individual runs independently on a processor, it may use a specific local hill-climbing method. This feature will be important for problems, where the efficiency of a particular hill-climbing method depends on the problem instance.

The PGA can be described as a parallel search with information exchange between neighbors in the space. Because the neighborhoods overlap, a *diffusion process* takes place. In the PGA all decisions are made by the individuals themselves. Therefore the *PGA is a totally distributed algorithm without any central control.*

There have been several other attempts to implement a parallel genetic algorithm. Most of the algorithms run k identical standard genetic algorithms in parallel, one run per processor. They differ in the linkage of the runs. Tanese [48] introduces two *migration* parameters: the *migration interval*, the number of generations between each migration, and the *migrationrate*, the percentage of individuals selected for migration. The subpopulations are configured as a binary n-cube. In the implementation of Cohoon [5] it is assumed that each subpopulation is connected to each other. The algorithm from Manderick et al. [30] has been derived from our PGA. In this algorithm the individuals of the population are placed on a planar grid and selection and crossing-over are restricted to small neighborhoods on that grid.

All but Manderick's algorithm use subpopulations that are densely connected. We have shown in [34] why restricted connections like a ring are better for the parallel genetic algorithm. All the above parallel algorithms do not use hill-climbing, which is one of the most important parts of our PGA.

Before we report applications of the PGA, we discuss the importance of spatial structures and collaboration in biology.

4 Co-evolution and Collaboration in Evolution

Collaboration is not part of evolution theories, it plays the dominant role in a separate field, today called *ecology*. Evolution theories try to understand how collaboration evolves and changes, whereas ecology models the existing collaboration in a habitat, trying to understand its influence on the size of the populations. A famous example are predator-prey cycles described in equation 4. In this section we discuss the interaction of populations in spatial distributions and its importance for the speed of evolution.

4.1 Darwin Revisited

First we will show that Darwin was aware of the importance of external conditions of an-organic nature like the geographic distribution. In the section “Circumstances favourable and unfavourable to Natural Selection” Darwin tries to describe the influence of intercrossing, isolation and number of individuals on the production of new organic forms. Darwin mentions that this is an extremely intricate subject. He argues very carefully, for example: ”For to ascertain whether a small isolated area or a large open area like a continent has been the most favourable for the production of new organic forms, we ought to make comparisons within equal times; and this we are incapable of doing’”.

Simulation makes such a comparison possible. It is of course impossible to simulate the real evolution of nature, so we have to find an artificial environment which is nevertheless complex enough to model the important aspects of evolution. As a first step we have decided to simulate an artificial population where each individual plays a two-person game against the other individuals. We have selected the Iterated Prisoner’s Dilemma IPD, because it is surprisingly complex. The simulation will be discussed in Section 5.

After a lot of reasoning Darwin arrives at the following conclusion. “I conclude that a large continental area, which will probably undergo many oscillations of level, and which consequently will exist for long periods in a broken condition, will be the most favourable for the production of many new forms of life, likely to endure long and spread widely.” Darwin argues as follows:

In a large continent, there is severe competition. This leads to the extinction of over-specialized species. But it is highly improbable that something new arises on a large continent. This happens much easier on small islands. But if the islands are isolated for a long time, then over-specialized forms will develop.

So Darwin postulates, that the islands should reconvert to a large continent. There will again be severe competition eliminating the specialized forms. This briefly sketches Darwin's derivation of his hypothesis. The interested reader is highly recommended to read the above mentioned chapter in Darwin's book. I found this conjecture so important that I have named it *Darwin's continent-island cycle* conjecture.

Mainstream biological science seems not to have noticed or to have deliberately neglected this section in Darwin's book. Darwin's evolution model is a non-equilibrium model, whereas all the popular Darwinian, Neo-Darwinian and Synthesis theories are equilibrium models.

Darwin's arguments in favor of the continent cycle can also, with some minor changes, be applied to other areas like the *invention of successful scientific ideas* or the *efficient organization of companies*. Take the organization of companies as an example. If the market (the environment) is stable, a large centralized company with severe internal competition is most effective. If the company has to adapt to a changing market, the large company should be subdivided into small companies which can adapt much faster.

4.2 Spatial Population Structures in Evolution Theories

Several researchers in biology have tried to investigate the importance of spatial population structures for evolution - without ever referring to Darwin. Space is an important element of the *shifting balance theory* of evolution proposed by Wright [55]. He argued that the best way to avoid a species being hung up on a low fitness peak is to have the population broken up into many nearly isolated subpopulations. Wright's theory has three phases [56]. *Phase 1* consists of the *differentiation* of innumerable small local populations by more or less random processes that occasionally lead to higher peaks. *Phase 2* is the occupation of higher peaks by *local mass selection*. *Phase 3* is the *diffusion* of these successful subpopulations throughout the species, followed by the appearance of still more successful centers of diffusion at points of contact. Then the whole process starts again.

Fisher [11], in contrast, argued that no such theory is needed. In a highly multidimensional fitness surface, the peaks are not very high and are connected by fairly high ridges, always shifting because of environmental changes. According to Fisher, the analogy is closer to waves and troughs in an ocean than to a static landscape. Alleles are selected because of their average effects, and a population is unlikely to be ever in such a situation that it can never be improved by direct selection based on additive variance.

The difference between these two views is not purely mathematical, but physiological. Does going from one favored combination of alleles to another often necessitate passing through genotypes that are of lower fitness? Fisher argued that evolution typically proceeds in a succession of small steps, leading eventually to large differences by the accumulation of small ones. According to this view, the most effective population is a large panmictic one in which statistical fluctuations are slight and each allele can be fairly tested in combination with many others alleles. According to Wright's view, a more favorable structure is a large population broken up into subgroups, with migration sufficiently restricted (less than one migrant per generation) and size sufficiently small to permit appreciable local differentiation.

Four different models for spatially structured populations have been investigated mathematically

- the one-island model
- the island model
- the stepping stone model
- the isolating by distance model

In the one-island model, an island and a large continent are considered. The large continent continuously sends migrants to the island. In the island model, the population is pictured as subdivided into a series of randomly distributed islands among which migration is random.

In the stepping-stone model migration takes place between neighboring islands only. One and two dimensional models have been investigated. The isolation by distance model treats the case of continuous distribution where effective demes are isolated by virtue of finite home ranges (neighborhoods) of their members. For mathematical convenience it is assumed that the position of a parent at the time it gives birth relative to that of its offspring when the latter reproduces is normally distributed.

Felsenstein [9] has shown that the isolating by distance model leads to unrealistic clumping of individuals. He concluded, that this model is biologically irrelevant. There have been many attempts to investigate spatial population structures by computer simulations, but they did not have a major influence on theoretical biology. A good survey of the results of the different population models can be found in [10]. Population models with oscillation like Darwin's continent-island cycle have not been dealt with.

The issue raised by Wright and Fisher is still not settled. Phase 3 of Wright's theory has been recently investigated by Crow [6]. He concludes: "The importance of Wright's shifting-balance theory remains uncertain, but we believe whatever weaknesses it may have, they are not in the third phase."

The problem of spatial population structures is now reappearing in the theory of genetic algorithms. The plain GA is based on Fisher's model. It is a well known fact, that the GA suffers from the problem of premature convergence. In order to solve this problem, many genetic algorithms enforce diversification explicitly, violating the biological metaphor. A popular method is to accept an offspring only if it is genetically more than a certain factor different from all the members of the population.

Our parallel genetic algorithm PGA tries to introduce diversification more naturally by a spatial population structure. Fitness and mating is restricted to neighborhoods. In the PGA we have implemented the *isolation by distance model and the stepping stone model*. The three phases of Wright's theory can actually be observed in the PGA. But the relative importance of the three phases are different than Wright believed. The small populations do not find better peaks by random processes. The biggest changes of the population occur at the time after migration between the subpopulations. Recombinations between immigrants and native individuals occasionally lead to higher peaks which were not found by any of the subpopulations during isolation. This behavior can easily be demonstrated in the application function optimizationfunction optimization (see [34] for details). We can therefore state the following observation.

The creative forces of evolution take place at migration and few generations afterwards. Wright's argument that better peaks are found just by chance in small subpopulations is wrong.

In our opinion the most important part of Wright's theory is what Wright postulated as "the appearance of still more successful centers of diffusion at points of contact". The difference of the evolution in a large continent and small isolated islands, has been recently investigated by [42].

We believe that static fitness functions cannot model natural evolution. In a real environment the fitness of an individual depends on the outcome of its interactions with other organisms in the environment. The fitness cannot be specified in advance. Therefore we used for a simulation of complex spatial population structures an evolutionary game.

5 The Iterated Prisoner'S Dilemma as an Evolutionary Game

In our artificial ecology the interactions of the individuals are modeled by a game. The fitness of the individual is the sum of the payoffs the individual gets during its lifetime. We have chosen the Iterated Prisoner's Dilemma (IPD), because it has been investigated from a number of different viewpoints.

The major emphasis of our research is on methodological questions for at least two reasons. First we believe that methodological questions are of utmost importance in a scientific field where it is almost impossible to compare simulation results with actual experiments. Second, a convincing simulation to support or disprove Darwin's continent cycle theory would need a tremendous effort.

Over its 60-year lifespan, the Iterated Prisoner's Dilemma has been one of the most frequently studied phenomena in economics, political science, sociology and psychology (see Axelrod [1] for a survey). The basic prisoner's Dilemma is a two-person game, with each player having a choice of either cooperating (C) or defecting (D). A typical set of payoffs is presented below.

Move	C	D
C	3/3	0/5
D	5/0	1/1

Given these payoffs, it is easily shown that mutual defection is the only Nash equilibrium. Of course, the intrigue of the Prisoner's Dilemma is that this unique equilibrium is Pareto inferior to the mutual cooperation outcome. If the basic Prisoner's Dilemma is iterated, the resulting super game is an Iterated Prisoner's Dilemma IPD. If the number of iterations is a known finite number, then a simple backward induction argument implies that the only equilibrium is mutual defection in every round. However, if the game is repeated a finite, but unknown number of times, then cooperative behavior can theoretically emerge.

The *ecological* approach to experimental games has added another dimension to the study of conflict and cooperation in societies. John Maynard Smith [46] introduced the *evolutionary game theory*, where the games are played by a population of individuals. The higher the payoff of an individual, the more offspring he will get. In this manner the most effective strategies survive. A strategy is called *evolutionary stable* [46], if it cannot be invaded by a single mutant strategy.

The theory assumes that the strategies are not changed during the course of evolution. In our simulations the strategies are coded by genes. The strategies are constantly changed by the parallel genetic algorithm, which uses mutation and crossing-over for generating offspring.

5.1 The Simulation of Spatial Structures Using the Iterated Prisoner's Dilemma

There have been many attempts to investigate the IPD with genetic algorithms. The first simulation was performed by Axelrod [2]. Axelrod considered strategies where the moves are based on the game's past three-move history. The major focus of Axelrod's study was on strategies evolving against a fixed environment. Each individual played against eight representative strategies. Marks [31] extended the investigation to *bootstrap* evolution, where the individuals play against each other. Miller [33] used finite automata to represent strategies. Furthermore he investigated the effect of informational accuracy on the outcome of the simulation. All three researchers used the plain genetic algorithm for evolving the population. They have been interested in equilibrium states and "optimal" strategies. We concentrate on the evolution of the behavior of the total population.

The PGA has been extended to simulate different population structures. The major enhancements of the PGA to the plain genetic algorithm are the spatial population structure, the distributed selection and the local hill-climbing. The individuals are active. They look for a partner for mating in their neighborhood. The partner is chosen according to the preference of the individuals. The best individual in a neighborhood has the chance to get as many offspring as the global best individual of the population. The PGA therefore has a very "soft" selection scheme. Each individual has the chance that on average 50% of its genes are contained in the chromosome of an offspring. The offspring replaces the parent. In order not to complicate the simulations the individuals are not allowed to improve their fitness by learning. This means their strategy is fixed during their lifetime.

We now turn to the problem of genetic representation of strategies.

5.2 The Genetic Representation

There are at least two obvious ways to represent strategies as a genetic chromosome, one is based on a simple table lookup, the other on a finite automaton. We will discuss deterministic table lookup strategies in this paper. A *k*-lookback strategy can be defined as a mapping of the outcome of the last *k* moves into a new move. In the simplest case of just looking one play back, a strategy can be defined by four entries in a table symbolizing the four possible moves of the last game - DD,DC,CD,CC. In addition two bits are necessary to specify the first move. The genetic representation of one-lookback thus consists of six bits. This gives 2^6 different genotypes. Three popular strategies are given below

						strategy
C	*	*	*	C	C	ALL-C
D	*	D	D	*	*	ALL-D
C	*	D	C	D	C	TIT-FOR-TAT

The sign * denotes that the allele on this locus does not have any influence on the performance of the strategy. The ALL-C strategy in row one is defined as follows. The player starts with C, then only two outcomes are possible, CD or CC. In both cases the player plays C. The outcomes DD and DC are not possible, therefore the entries in these columns are irrelevant. Altogether there are twelve different bit-strings which define an ALL-C strategy. The problem of this straightforward genetic representation is that we have a distinction between the *representation* and the *interpretation*. The program which interprets the representation is not part of the genetic specification and therefore not subjected to the evolution process.

But we have a clear distinction between genotype, phenotype and behavior. The genotype is mapped into some phenotype, the phenotype together with the environment (in our case the other phenotypes) defines the strategy. Let us take the famous TIT-FOR-TAT as an example. In TIT-FOR-TAT the player makes the move the opponent made the game before. In an environment where only C is played, TIT-FOR-TAT cannot be distinguished from an ALL-C player. A different behavior can only be recognized if there exists an individual who occasionally plays D.

The mapping from genotype to phenotype is many-to-one. This makes a behavior oriented interpretation of a given genetic representation very difficult. There exist no simple structure of the genotype space. The Hamming distance between two ALL-C genetic representations can be as large as four, whereas the Hamming distance between two very different strategies like ALL-C and ALL-D can be as small as one. An example is shown below

						strategy
C	C	D	D	C	C	ALL-C
C	C	D	D	C	D	ALL-D

If we assume that the genetic operators *mutation* and *crossing-over* uniformly explore the genotype space, then strategies like ALL-C and ALL-D will have a much higher chance to be generated than other strategies which are less often represented. The genetic search is therefore biased by the genetic representation. We believe that this effect is not a shortcoming of the chosen representation, but that this feature models real life evolution. The evolution has always to work within the constraints it creates for itself.

The complex mapping between genotype and phenotype makes it difficult to estimate the outcome of a genetic operator. For example, a winning strategy may be crossed with a losing strategy, giving in most cases a new strategy. An ALL-D strategy which is crossed-over with an ALL-C strategy gives with probability 0.2 ALL-D and with probability 0.2 ALL-C. With probability 0.6 we get a strategy which is different from the strategies of the parents.

We believe that in our artificial ecology the crossover operator is too disruptive compared to real evolution. The same problem occurs if the genetic representation is based on a finite automaton. In order to solve this problem we have to find a genetic representation which is based on a more complex genetic machinery than simple bit-strings. It is outside the scope of this paper to discuss this genetic machinery. We only want to mention that we have to incorporate some ideas of genetic models of self-reproduction proposed already in the 60's.

The influence of spatial population structures is independent of the genetic representation, therefore we will concentrate on this subject.

5.3 Mathematical Analysis of Structured Populations in Evolutionary Games

Before we discuss some of the simulation results in detail we want to show by a simple analysis how a spatial population structure influences the development of strategies. For simplicity we assume that we have a population, consisting of *inhabitants* playing strategy I and *invaders* playing strategy J. Let $0 < s < 1$ be the proportion of invaders. We assume that s is very small. Furthermore the invaders are clustered. We model this fact by a clustering factor $0 < k \leq 1/s$.

Let $P(I,J$ denote the payoff of an individual playing strategy I against an individual playing strategy J. After invasion the fitness of the inhabitants $F(I)$ is given by the outcomes of plays with each other and against the invaders. Computing the fraction of each of the games we obtain

$$F(I) = (1 - s * \frac{1-ks}{1-s}) * P(I,I) + s * \frac{1-ks}{1-s} P(I,J) \tag{6}$$

The fitness of the invaders is given by

$$F(J) = (1-ks) * P(J,I) + ks * P(J,J) \tag{7}$$

We see that for $k = 0$ the invaders play against the inhabitants only, the case $k = 1$ gives the panmictic population normally considered in the theory of evolutionary games. Here the plays are performed according to the frequency of the actors. In the case of $k > 1$ we have a clustering effect. The players play more often within their groups (inhabitants, invaders). For $k = 1/s$ the effect is most dramatic. The mixed terms with $P(I,J)$ vanish, thus the invaders and the inhabitants play within their group only. This is a very crude model of a structured population, but it can be used to show some important points.

A strategy is called *collective stable* if no strategy can invade it. A new strategy is said to *invade* if the newcomer gets a higher score than the native strategy, this means that $F(I) < F(J)$. In order to obtain a simple formula, we assume that s is small approximate $F(I)$ by $P(I,I)$. Thus the small number of plays between inhabitants and invaders is ignored. We get

$$P(I,I) < (1-ks)P(J,I) + ks * P(J,J) \tag{8}$$

It is now easily seen that even ALL-C can withstand the invasion of ALL-D, if there is a strong preference for each strategy to play only against each other. With our payoff values we obtain that ALL-C will not be invaded by ALL-D if $k > 0.5s^{-1}$. But also the other invasion is possible. ALL-C can invade an ALL-D population as long as they "stick together". This means they play, even after the invasion, much more against each other than against ALL-D. [4]

In a one-dimensional spatial population structure with fixed neighborhoods the situation is more difficult. The contest between the strategies happens at the boundary of the neighborhoods, whereas the individuals in the interior play only against members of their own group. In this spatial structure the success of the invasion is therefore totally determined by the outcomes at the boundary.

It is almost impossible to investigate realistic spatial population structures by analytical methods, one has to use simulations. This was first done by Axelrod ([1], pp. 158-168). Axelrod investigated a simple 2-D structure where each player had four neighbors. The selection was very strong. If a player had one or more neighbors which had been more successful, the player converted to the strategy of the most successful of them. Axelrod's major conclusion was that mutual cooperation can be sustained in a (not too highly connected) territorial system at least as easily as it can be in a freely mixing system. We will extend Axelrod's work. First, different population structures are compared and second, the strategies evolve controlled by the genetic algorithm.

5.4 Simulation Results

In our simulation we have investigated the following population structures

[4] If we set $p = ks$ then the above inequality is identical to Axelrod's *p-cluster invasion* ([1],p.212).

- a small panmictic population
- a large panmictic population(500 individuals)
- a one-dimensional population (ring structure with four neighbors)
- the continent-island cycle: a cycle between ten islands populations and a panmictic population
- a competition between five population structures

In a panmictic population each individual plays against each other, in a spatial population structure the individuals play only against their neighbors. Most of the experiments have been done with a small population of 50 individuals. Detailed simulation results can be found in [3]. We outline in this paper only the major facts supporting or disproving Darwin's argument. In our simulations we used 2-lookback strategies. They can be coded by 20 bits. This gives 2^{20} different genotypes.

The figures show individual runs which are "representative". Each individual experiment has been repeated 10 times. The results are qualitative as described. The figures are what we believe "average" runs. Because of the stochastic nature of the individual runs (especially the runs with small populations) it makes no sense to average the fitness over the 10 different runs. A small panmictic population for instance occasionally changes to a non-cooperative behavior for a certain time. The time when this will happen cannot be predicted.

For a qualitative investigation we used the average fitness of the population. Batz [3] has used more difficult measures like the distribution of classes of strategies. Given the pay-off table theoretical research indicates that strategies like TIT-FOR-TAT (TFT) are evolutionary stable, leading to an average fitness of three.

The difference between a large panmictic population and a small one is shown in Figures 1 and 2. The simulation of the large population started with three predefined strategies - 5 ALL-D, 490 ALL-C and 5 TFT. The population heads first to non-cooperative behavior. Then ALL-D is beaten by TFT. The population arrives at cooperation in generation 45.

Similarly the small population started with 5 ALL-D, 40 ALL-C and 5 TFT. The small panmictic population oscillates, but is also heads first to non-cooperative behavior. It arrives at cooperation at generation 65.

The result of this experiment can be explained mathematically. The initial fitness of the strategies can easily be computed. In the small population the fitness of ALL-D is given by

$$F(ALL-D) = (4+200+5)/49 = 4.27$$

Similarly we obtain $F(TFT) = 2.84$ and $F(ALL-C) = 2.69$. Thus ALL-D first increases by eliminating ALL-C. Then ALL-D is conquered by TFT. For the large population we get $F(All-D) \approx 5, F(TFT) \approx 3, F(ALL-C) \approx 3$. The difference of the fitness between ALL-D and TFT is larger in the large population. This explains the rapid increase of ALL-D in the large population at the beginning.

The ring population oscillates as shown in Figure 3. The selection scheme of the PGA is too soft for this population structure. In order to implement a higher selection pressure we introduced an acceptance test of the offspring. In the first scheme the offspring replaced the parent only if it won the IPD against the parent.

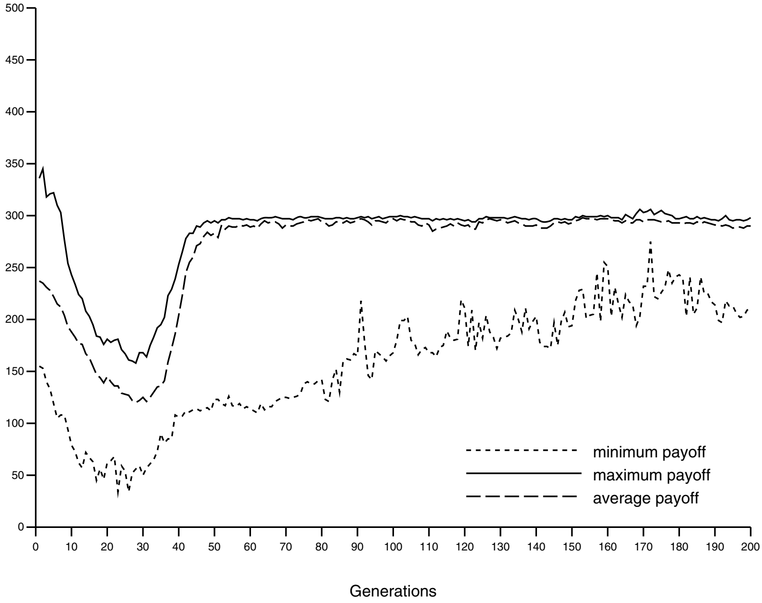


Fig. 1 Large panmictic population

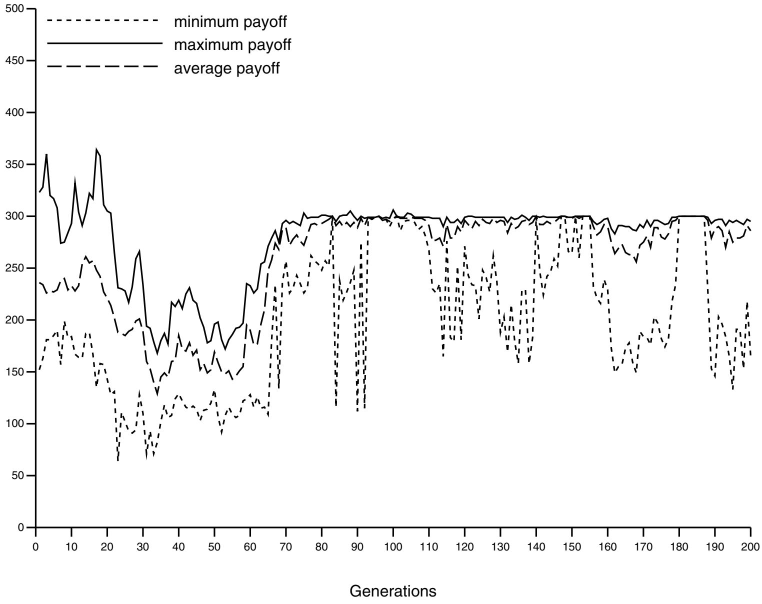


Fig. 2 Small panmictic population

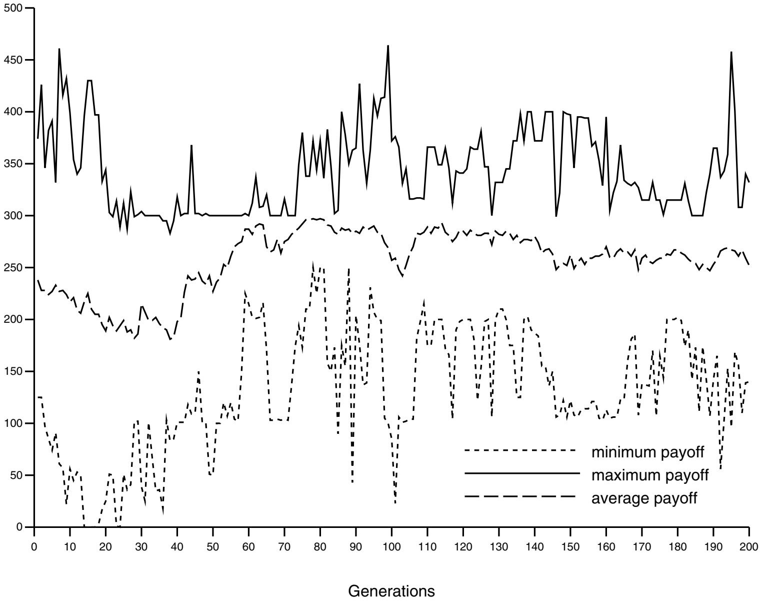


Fig. 3 Ring population

The effect was dramatic. Now the population always settled on non-cooperative behavior. The situation changed with our second scheme. This extension we called the *family game*. Each mating produces two offspring. After the mating the family consisting of the two parents and the two offspring plays an IPD tournament. The winner replaces the parent. With this selection scheme the population settled on cooperative behavior.

The explanation of this result is simple. In the IPD non-cooperative strategies can be eliminated if the cooperative individuals stick together. In a single contest, ALL-D can never be beaten. It is outside the scope of this paper to compare the family game with *kin selection* proposed in sociobiology [54].

In Figure 4 the continent-island cycle is shown. One easily recognizes the cycle (20 generations island, 20 generations continent). During the continent phase the variance is reduced, during the island phase it is increased.

In Figure 5 the average fitness of the population is shown for five different population structures. The simulation started with a homogeneous ALL-D population. We investigated whether the populations will change to cooperation. We see that the population which is subjected to the continent-island cycle is first to arrive at cooperation. This result was consistent in ten runs. A closer analysis of the strategies showed that the winning cooperative strategies are not naive like ALL-C, but they resemble TIT-FOR-TAT.

In a further set of experiments we changed the game during the course of the simulation, for instance we changed the IPD to the chicken game. The spatial structured

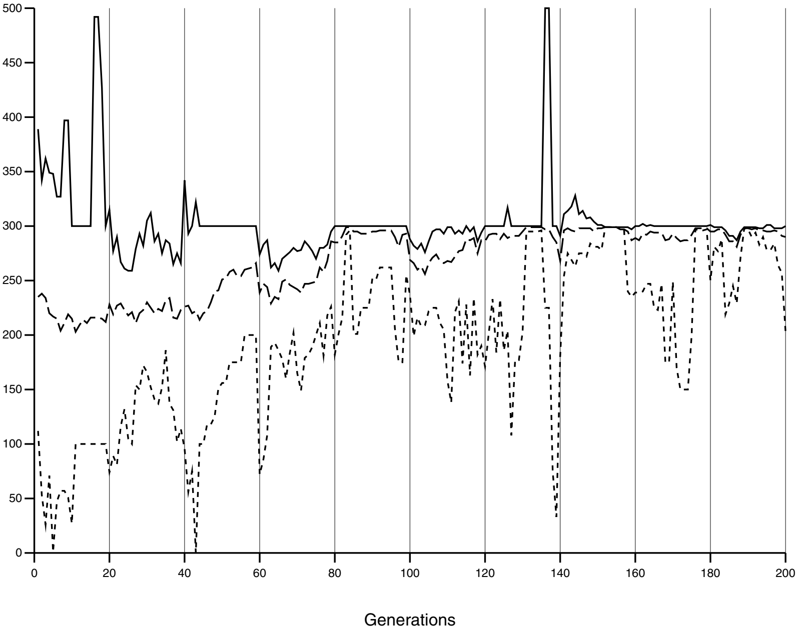


Fig. 4 Continent-island cycle

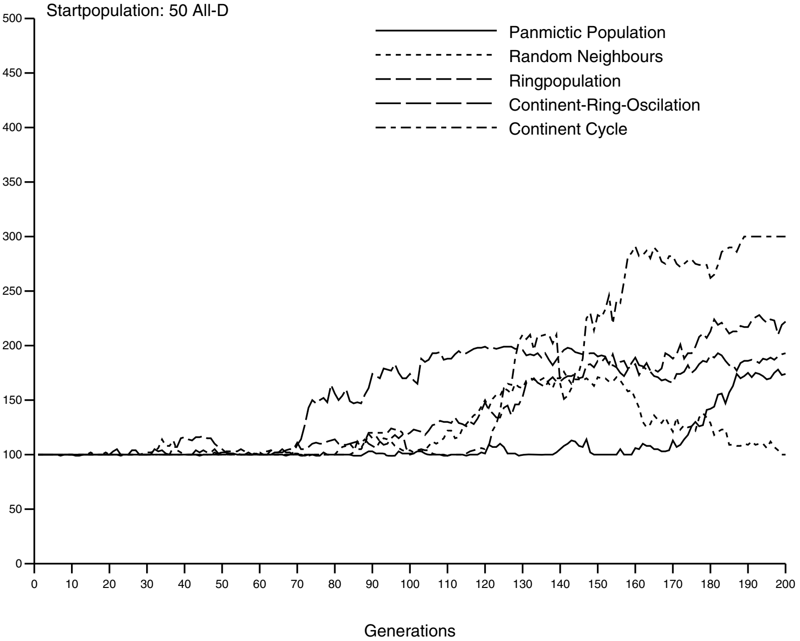


Fig. 5 Starting with defecting

populations adapted much faster to the new game than a large panmictic population. This is one of the extensions that have been already proposed by Axelrod for investigation ([1], p.221).

5.5 The Punctuated Equilibrium Theory

In our simulation, we observe rapid changes in fitness, followed by long intervals of small fluctuations. This offers some support for a view of evolution called *punctuated equilibrium* [16]. This concept was controversial in evolutionary biology for some time, but it has now many supporters. Let us describe the theory and its historical context.

During the years of Darwin's apprenticeship in science there has been an intense conflict in geological circles between adherents of *rapid* and *gradual* changes of the fossil record. There is no doubt that Darwin strictly favored gradualism in his writing.

This posed a dilemma to paleontologist. In order to favor gradual changes by natural selection, they had to view the empirical data as so incomplete that the process of natural selection cannot be observed.

Gould and Eldredge abandoned this view by rejecting gradualism. They observed in the history of most fossil species two features particularly inconsistent with gradualism:

- **Stasis:** Most species exhibit no directional change during their tenure on earth. They appear in the fossil record looking much the same as when they disappear; morphological change is usually limited and direction less.
- **Sudden appearance:** In any local area, a species does not arise gradually by the steady transformation of its ancestors; it appears all at once and 'fully formed'.

This behavior they called punctuated equilibrium. It explains the fossil record. The evolutionary forces responsible for this behavior are still controversial. In all simulations the time of changes (indicated by a change in average fitness) is small compared to the time of the equilibrium.

We next turn to a different application of the PGA, the solution of difficult combinatorial problems.

6 Combinatorial Optimization by the PGA

Most applications of genetic algorithms are optimization problems. Here a static fitness function is used derived from the optimization problem. The search strategy of the PGA is driven by three components - the spatial population structure, the crossover operator and the hill-climbing strategies. The spatial structure has been discussed in the previous section. In this section we concentrate on the problem dependent aspect, the crossover operator and the local hill-climbing.

There have been attempts to "prove" that genetic algorithms make a nearly optimal allocation of trials. This result is called the "Fundamental Theorem of Genetic Algorithms" (Goldberg [13]) We have shown already in [34] that the above claim is only valid for simple optimization problems. In fact, in [37] we have proven a correct schema theorem, based on Boltzmann selection and our Estimation of Distribution family of algorithms [29, 36].

The search strategy of a genetic algorithm can be explained in simple terms. The crossover operator defines a *scatter search* [12] where new points are drawn out of the area which is defined by the old or "parent" points. The more similar the parents are, the smaller will be the sampling area. Thus crossing-over implements an adaptive step-size control.

But crossing-over is also exploring the search space. Let us assume that the combinatorial problem has the *building block feature*. We speak of a building block feature if the substrings of the optimal solutions are contained in other good solutions. In this case it seems a good strategy to generate new solutions by patching together substrings of the old solutions. This is exactly what the crossover operator does.

6.1 The Traveling Salesman Problem

The major difficulty for applying the PGA to combinatorial problems is to define a crossover operator which creates valid solutions i.e. solutions which fulfill the constraints of the problem. We will explain this problem first with the TSP.

THE TRAVELING SALESMAN PROBLEM

OPT 1 (TSP) *Given are n cities. The task of the salesman is to visit all cities once so that the overall tour length is minimal.*

This problem has been investigated in [35, 14, 15, 34] with the PGA. The genetic representation is straightforward. The gene at locus *i* of the chromosome codes the edge (or link) which leaves city *i*. With this coding, the genes are not independent from each other. Each edge may appear on the chromosome only once, otherwise the chromosome would code an invalid tour. A simple crossing-over will also give an invalid tour. This is the reason why this simple genetic representation has not been used in genetic algorithms. The researchers tried to find a more tricky representation in order to apply a simple crossover operator.

We take the opposite approach. We use a simple representation, but an intelligent crossover operator. Our crossover operator for the TSP is straightforward. It inserts part of chromosome A into the corresponding location at chromosome B, so that the resulting chromosome is the most similar to A and B. A genetic repair operator then creates a valid tour.

We call our crossover operator MPX, the maximal preservative crossover operator. It preserves sub-tours contained in the two parents. The pseudocode is given below.

```
PROC  crossover (receiver, donor, offspring)
    Choose position $0 <= i < nodes$ and length $b_{low} <= k <= b_{up}$ randomly.
    Extract the string of edges from position $i$ to position $j = (i+k)\,\mathrm{MOD}\, nodes$ from the
    mate (donor). This is the crossover string.
    Copy the crossover string to the offspring.
    Add successively further edges until the offspring represents a valid tour.
    This is done in the following way:
        IF an edge from the receiver parent starting at the last city in the offspring is possible (does not violate a valid tour)
        THEN  add this edge from the receiver
        ELSE  IF an edge from the donor starting at the last city in the offspring is possible
        THEN  add this edge from the donor
        ELSE  add that city from the receiver which comes next in the string, this adds
        a new edge, which we will mark as an implicit mutation.
```

We want to recall, that in the PGA the crossover operator is not applied to all TSP configurations, but only to configurations which are a local minima. Our local search is a fast version of the 2-opt heuristic developed by Lin [28]. It is a 2-opt without checkout. It gives worse solutions than 2-opt, but the solution time scales only linearly with the number of cities.

We have later found that the efficiency of the PGA increases with the quality of the local search. But the major goal of the PGA work on the TSP was to investigate the problem independent aspects i.e. the population structure and the selection schedule. Therefore many generations were needed, which could only be obtained by a fast local search method.

We turn to a popular benchmark problem, the ATT-532 problem solved to optimality in [41]. The PGA with a population size of 64 and truncated 2-opt as local search method got a tour length of 0.10% above optimal in ten runs of $t = 1080s$ (1000 generations,15000 local searches) on a 64 processor system, the average final tour length was 0.19% above optimal [14]. This is a substantial improvement over the results in [49] for genetic 2-opt search. It demonstrates the robustness of the parallel genetic algorithm. The PGA finds good solutions with a simple local search also.

This implementation had some influence in the development of heuristics for the TSP. There is a section about the different PGA implementation in Johnson and McGeoch's seminal paper [26].

We will compare our heuristic with a very fast and efficient heuristic proposed by Johnson [24]. It is called *iterated Lin-Kernighan* search. In his implementation a new start configuration is obtained by an unbiased 4-opt move of the tour at hand. Then a new L-K search is started. If the search leads to a tour with a smaller tour length, the new tour is accepted.

Johnson reports the following results. In time $t = 2700s$ (500 L-K searches) the optimal tour (length 27686) was output 6 of 20 IterL-K runs, the average final tour-length was 0.05% above optimal. Multiple L-K runs gave much worse results. A single L-K run averages 0.98% above optimal in time $t = 120s$. 100 L-K runs gave a tour length of 0.25% above optimal. It needed 20000 L-K runs ($t = 530 hours$) to obtain a tour of length 27705.

Why is IterL-K more efficient than independent L-K runs? The success of IterL-K depends on the fact that good local L-K minima are clustered together and not randomly scattered. The probability to find a good tour is higher nearby a good tour than nearby a bad tour.

We have shown in [34] that 2-opt local minima are clustered. Furthermore we could show the following relation: The better the solutions are, the more similar they are. This relation is the reason for the success of Johnson's IterL-K. The relation holds, if the problem has the *building block feature*, which is necessary for the success of the crossover operator of our genetic algorithm.

Iterated hill-climbing needs a fine tuned mutation rate to get with high probability to the attractor region of a new local minimum. In the TSP case Johnson found that a simple 4-opt move is sufficient. In other combinatorial problems it is more difficult to find a good mutation rate and a good local heuristic like the Lin-Kernighan search for the TSP. Therefore we share the opinion of Johnson that the TSP is in practice much less formidable than its reputation would suggest [24]. An in-depth evaluation of heuristics for the solution of large TSP problems can be found in [26].

6.2 The Graph Partitioning Problem

We will now turn to another combinatorial problem, the graph partitioning problem GPP. Here the Lin-Kernighan heuristic is not as good as in the TSP case. We will show that the genetic search is very efficient for this problem. The major obstacle is to find a suitable crossover operator.

THE GRAPH PARTITIONING PROBLEM

The m graph partitioning problem (m-GPP) is a fundamental problem which arises in many applications. The GPP is to divide a given graph into a number of partitions (m) in order to optimize some criterion e.g. to minimize the number of edges between partitions. More formally:
Let a graph $G = (V, E, w)$ be given. $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes, $E \subseteq V \times V$ is the set of edges and $w : E \mapsto I\!\!N$ defines the weights of the edges.
The m-GPP is to divide the graph into m disjunct parts, such that some optimization criteria will be fulfilled. In this paper we will consider the following optimization criteria:

OPT 2 (m-GPP) *Let $\mathscr{P} = \{P_1, ..., P_m\}$ be a partition. Let $\mathscr{G} = (g_1 g_2 ... g_n)$ denote the partition to which the nodes belong $(1 \leq g_i \leq m)$. Then we look for*

$$\min_{\mathscr{P}} \sum_{\substack{1 \leq i < j \leq n \\ g_i \neq g_j}} w_{ij}$$

such that $\sigma(P)$ is minimal.

$\sigma(P)$ is defined as

$$\sigma^2(P) = \frac{1}{m}\sum_{i=1}^{m}|P_i|^2 - (\frac{1}{m}\sum_{i=1}^{m}|P_i)^2$$

In order to solve the GPP, we have to define the genetic representation and the genetic operators. In the simplest representation, the value (allele) g_i on locus i on the chromosome gives the number of the partition to which node v_i belongs. But this representation is highly degenerate. The number of a partition does not have any meaning for the partitioning problem. An exchange of two partition numbers will still give the same partition. All together m! chromosomes give the same fitness value.

$$F(\mathscr{G}) = \sum_{\substack{1\le i<j\le n \\ g_i \neq g_j}} w_{ij}$$

All m! chromosomes code the same partitioning instance, the same "phenotype". The genetic representation does not capture the structure of the problem. We did not find a better genetic representation, so we decided that the crossover operator has to be "intelligent". Our crossover operator inserts complete partitions from one chromosome into the other, not individual nodes. It computes which partitions are the most similar and exchanges these partitions. Mathematically speaking, the crossover operator works on equivalence classes of chromosomes.

Figure 6 shows an example. The problem is to partition the 4×4 grid into four partitions.

The crossover operator works as follows. Partition 2 has to be inserted into $\mathscr{B}$. The crossover operator finds, that partition 4 of $\mathscr{B}$ is the most similar to partition 2 in $\mathscr{A}$. It identifies partition 2 of $\mathscr{A}$ with partition 4 of $\mathscr{B}$. Then it exchanges the alleles 2 and 4 in chromosome $\mathscr{B}$ to avoid the problems arising from symmetrical solutions. In the crossover step it implants partition 2 of chromosome $\mathscr{A}$ into $\mathscr{B}$.

After identifying all gene-loci and alleles which lead to a non-valid partition a repair operator is used to construct a new valid chromosome. Mutation is done after the crossover and depends on the outcome of the crossover. In the last step a local hill-climbing algorithm is applied to the valid chromosome.

For local hill-climbing we can use *any* popular sequential heuristic. It should be fast, so that the PGA can produce many generations. In order to solve very large problems, it should be of order $O(n)$ where n is the problem size. Our hill-climbing algorithm is of order $O(n^2)$, but with a small constant. In order to achieve this small constant, a graph reduction is made. The general outline of our hill-climbing algorithm is as follows:

Local search for the GPP

1. Reduce the size of the graph by combining up to r nodes into one hyper-node
2. Apply the 2-opt of Kernighan and Lin [27] to the reduced Graph. For the GPP it is defined as follows:

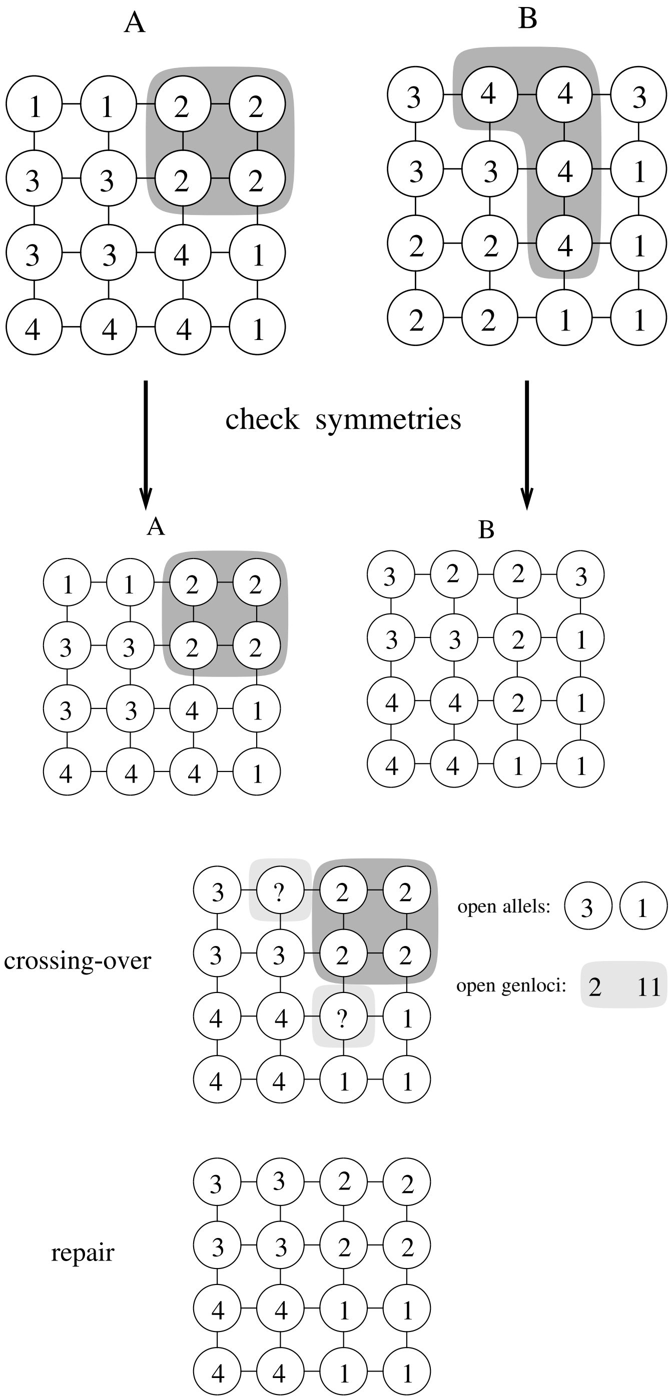


Fig. 6 The crossover operator for the m-GPP

a. Select two hyper-nodes
b. Test if an exchange of this hyper-nodes gives a lower fitness
c. If this is the case, exchange the nodes
d. Terminate, if no exchange can be made over the set of nodes

3. Expand the resulting graph
4. Create a valid partition
5. Apply a further local hill-climbing algorithm to the valid partition

Step2 and step5 are of order $O(n^2)$. The constant is smaller than doing 2-opt search on the original string. The above local search is only done for the initial population. After the first generation we use a still faster search. We apply 2-opt only to the nodes which have connections to outside partitions.

The general m-GPP problem has been rarely studied. More popular is the the bi-partitioning problem. A detailed study of the bi-partitioning problem can be found in [25]. In that paper random graphs and random geometric graphs up to 1000 nodes are used to compare different heuristics. We decided to make a performance analysis with real life graphs. Furthermore we are more interested in the general partitioning problem, not in the bi-partitioning problem. Detailed results can be found in [52]. Results for the bi-partitioning problem using our new Factorized Distribution Algorithm FDA can be found in [37].

We will give here the computational results for solving a large GPP benchmark problems. The problem is called *EVER918*. It is a 3-D graph which consists of 918 nodes and 3233 edges. It has to be partitioned into 18 partitions.

In Figure 7 we show the progress of the PGA for problem $EVER918$. The progress is typical. The best progress is made at the beginning, then it decays exponentially.

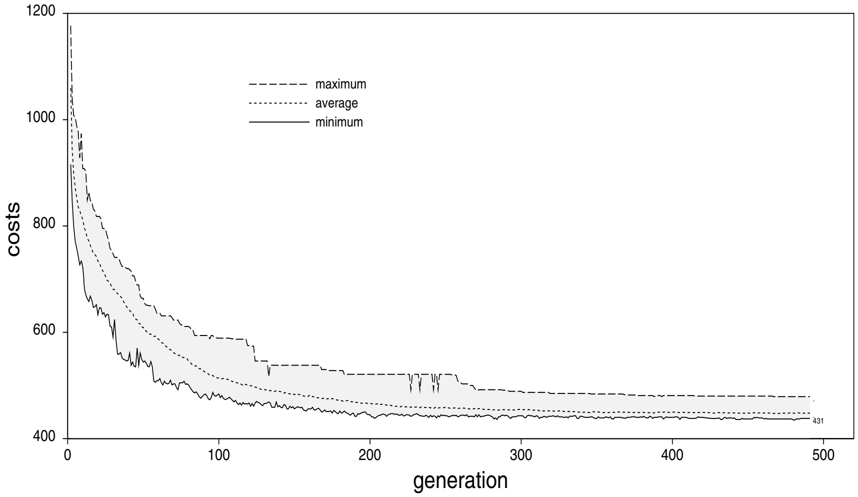


Fig. 7 Problem EVER918, 64 individuals

For EVER918 the PGA found the best solution computed so far [53]. Further investigations have indicated that it will be difficult to construct an efficient iterative Lin-Kernighan search for the m-GPP. First, the quality of an average L-K solution is bad for the GPP. Second, it is difficult to determine a good mutation rate which jumps out of the attractor region of the local minimum. This has been demonstrated in [53]. There the following relation was shown: the better the local minimum, the larger its attractor region.

In summary: The m-GPP problem is more difficult to solve than the TSP. The PGA got results which are better than other known heuristics.

7 Continuous Function Optimization by Competition

Optimization of multi-modal continuous functions was a notoriously difficult task for genetic algorithms. Sometimes good results have been achieved, but for many benchmark problems genetic algorithms have not been competitive with other methods. Therefore we developed an algorithm called the *Breeder Genetic Algorithm* BGA. The BGA uses a real-valued representation [38, 39]. Therefore a number of specialized mutation and recombination operators have been implemented. These depend on parameters. In order to automatically adapt the parameters, *competition* between subpopulations using different parameters has been implemented. Detailed discussions of the competition scheme can be found in [39, 45].

7.1 The BGA for Continuous Parameter Optimization

Let an unconstrained optimization problem be given on a domain $D \subset \Re^n$

$$\min(F(\mathbf{x})) \quad a_i \le x_i \le b_i \quad i = 1, \ldots, n \ . \tag{9}$$

The breeder genetic algorithm **BGA** was designed to solve the above problem [38]. The BGA uses a set of search strategies. An overview of search strategies based on recombination can be found in [51] and [50]. In [51] it was shown that a new recombination scheme called **fuzzy recombination** (FR) worked best as a breadth search. In this chapter we only describe the BGA mutation scheme and the BGA line recombination which uses also the mutation scheme.

BGA mutation $\mathsf{BM}(\rho, k, \nu)$

The BGA mutation scheme depends on the *mutation range* ρ, the *precision* k and a new parameter ν which gives the number of neighboring variables to be mutated. The standard BGA mutation ($\nu = 0$) randomly selects just one variable x_i.

Given x_i a new value z_i is computed according to

$$z_i = x_i + \rho_i \cdot \delta(k) \tag{10}$$

ρ_i is normally set to the domain of definition of variable x_i. $\delta(k)$ is a random variable which is computed as follows:

$$\delta(k) = \mathrm{sign}(\alpha) \cdot 2^{-k \cdot |\alpha|} \quad \alpha = \mathbf{U}(-1,1)$$

where $\mathbf{U}(u,v)$ denotes the uniform probability distribution with support $(v,w) \subset \Re$.

k is called the precision constant. The smallest absolute value of $\delta(k)$ is 2^{-k}, the highest value is 1.0. Therefore the step sizes of the BGA-mutation are contained in the interval $[\rho_i \cdot 2^{-k}; \rho_i]$.

The rationale of the BGA mutation scheme has been explained in [38]. An extension of the BGA-mutation is specified by its third parameter ν. Now additionally to the randomly chosen variable x_i also adjacent variables are modified. The higher the distance to i the smaller is the change.

$$z_{i-j} = x_{i-j} + \rho_{i-j} \cdot 2^{-j} \tag{11}$$

$$z_{i+j} = x_{i+j} + \rho_{i+j} \cdot 2^{-j} \tag{12}$$

$$\begin{aligned} &for \;\; j = 1, \ldots \nu \\ &and \; i - j > 0, \quad i + j \leq 0 \end{aligned}$$

where Δ is the value of $\delta(k)$ generated for x_i

The parameter ν defines the size of the neighborhood. It lies in the interval $[0;n]$. For the standard BGA-mutation we have $\nu = 0$. Note that the mutation step decreases exponentially starting from variable x_i. This is in accordance to the design rationale of the BGA mutation.

BGA line recombination $\mathsf{BLR}(\rho, k)$

The BGA line recombination uses components from both, mutation and recombination. It creates new points in a direction given by the two parent points. The placement of the point is done by the BGA mutation scheme. It works as follows: Let $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ be the parent strings with $\mathbf{x}$ being the one with better fitness. Then the offspring $\mathbf{z} = (z_1, \ldots, z_n)$ is computed by

$$z_i = x_i + \rho_i \cdot \delta(k) \cdot \frac{y_i - x_i}{\|\mathbf{x} - \mathbf{y}\|} \tag{13}$$

The BGA line recombination may generate points which are far from the given point x_i.

7.2 *Competition between Subpopulations*

In order to automatically adapt the parameters, we had the idea to use subpopulations each with a different set of parameters. The subpopulations with high fitness should increase, the ones with low fitness should decrease. Before implementing a scheme we looked if theoretical models could guide us.

Ecology deals with the interaction of subpopulations and species. Unfortunately even the analysis of the interaction between two species can be quite complicated, involving the effects of exterior and interior parameters. As a first approximation one may distinguish four basic situations — competition, predator-prey, symbiosis and host-parasite.

The most popular equations for analyzing the interaction of species are generalizations of the famous Lotka-Volterra equations. For two species they are as follows [8]

$$\frac{dN_1}{dt} = r_1 \cdot N_1 \left(1 - \frac{N_1}{K_1} - \alpha_{12}\frac{N_1}{K_2}\right) \tag{14}$$

$$\frac{dN_2}{dt} = r_2 \cdot N_2 \left(1 - \frac{N_2}{K_1} - \alpha_{21}\frac{N_1}{K_2}\right) \tag{15}$$

Here N_1, N_2 denote the population sizes of the two species, r_1, r_2 are the growth rates, K_1, K_2 the carrying capacities and $\alpha_{12}, \alpha 21$ the interaction coefficient. This equation has been studied intensively [8]. It is very useful for understanding the complex patterns which may arise by two interacting species. For a competition scheme to be implemented these equations cannot be used because the interaction coefficients cannot be specified in advance. In analogy to the above model the following model has been implemented.

7.3 The Basic Competition Model of the BGA

Our competition scheme requires a *quality criterion* to rate a group, a *gain criterion* to reward or punish the groups, an *evaluation interval*, and a *migration interval*. The evaluation interval gives each strategy the chance to demonstrate its performance in a certain time window. By occasional migration of the best individuals, groups which performed badly are given a better chance for the next competition. The sizes of the groups have a lower limit. Therefore no strategy is lost. The number of competing subpopulations (S) depends on the set of strategies used in the competition. Normally the number of groups is between 2 and 8.

Our **quality criterion** (Q) is based on the fitness of the best individual of the group. To avoid an inefficient oscillation of group sizes we use information about the last ω competitions for the evaluation. The vector $\mathbf{w} \in N^{\omega}$ provides the winners of the last ω competitions. $w_k \in \{1, \ldots, S\}$ contains the winner of the k-last competition.

The following formula describes the quality of group i. $k = 0$ denotes the current competition, $k = 1$ the previous one, etc. The time window ω is 10.

$$Q_i(\mathbf{w}) = \sum_{k=0}^{\omega-1} \begin{cases} (\omega - k)/\omega & : \quad i = w_k \\ 0 & : \quad i \neq w_k \end{cases} \tag{16}$$

The **gain criterion** (G) defines how the population size of each group is modified according to its quality. Normally, the size of the group with the best quality increases, the sizes of all other groups decrease. The following scheme increases the size of the best group (w_0) by the accumulated loss of the others. The loss of a group is proportional to its population size. The loss factor $\kappa \in [0;1]$ defines the rate of loss.

The change of the population sizes is computed from the following equations:

$$\Delta N_i = \begin{cases} \sum_{j=1,j\neq i}^{S} N_j^t \cdot \kappa & : \begin{array}{l} Q_i(\mathbf{w}) > Q_j(\mathbf{w}) \\ \forall j, j \neq i \end{array} \\ -N_i^t \cdot \kappa & : \ else \end{cases} \tag{17}$$

where N_i^t denotes the size of group i and S denotes the number of groups. The loss factor κ is normally set to 0.125.

The population size of each group of the next generation is given by:

$$N_i^{t+1} = \begin{cases} N_i^t + \Delta N_i & : \ N_i^t + \Delta N_i \geq N^{min} \\ N_i^{min} & : \ else \end{cases} \tag{18}$$

The size of the population is only reduced if it is greater than the minimal size N^{min}.

This gain criterion leads to a fast adaptation of the group sizes. Each group looses the same percentage of individuals.

The **evaluation interval** η and the **migration interval** θ are rather robust external parameters. Normally we set $\eta = 4$ and $\theta = 16$.

If one compares equations 17 with the generalized Lotka-Volterra equation 14 the following major difference can be observed. Our equations are linear whereas the Lotka-Volterra equations contain the nonlinear term $N_i \cdot N_j$. The reason for this difference is that the Lotka-Volterra equations model individual competition. If there are many predators and each one captures two preys, then the reduction of the preys depends on the number of predators. In contrast, our competition scheme evaluates whole groups by taking the best individual as evaluation criterion.

The current competition scheme seems appropriate in cases when the strategies used by the different groups differ substantially. Sometimes a competition model might be better where even the size of the total population may vary.

7.4 The Extended Competition Model

If search strategies differ very much they may also require a different population size to be efficient. It has been shown in [38] that mutation is most efficient in small populations whereas recombination needs a larger population size. This can be modeled by introducing growth rates which depend on the group.

In our implementation we introduced a *consumption factor* γ for each subpopulation. Biologically speaking, a consumption factor specifies the consumption of the limited resource by one individual of a species — the higher the consumption factors the lower the number of individuals which can be supported by that resource. We implemented this extension by introducing a *normalized population size* $\tilde{N}$.

$$\tilde{N}_i = \gamma_i \cdot N_i \tag{19}$$

The gain criterion of equation 17 is now applied to the normalized population sizes. The sum of the normalized population sizes remains constant because it is limited by the limited resource K.

$$\sum_{i=1}^{S} \tilde{N}_i = K \tag{20}$$

For $\gamma_i = 1.0$ for $i = 1, \ldots, S$ we obtain the basic model.In contrast to the basic model the sum of the real population sizes varies during a simulation. This extended competition scheme can be very effective for multi-modal problems where it is useful to locate the region of attraction of the global (or a good local) optimum by a breadth search and to do the fine adaptation by an exploring strategy afterwards. In this case the strategy performing breadth search gets a lower γ than the other strategy. So the total population size is high at the beginning when the breadth search works and low at the end when the fine adaption is done. Thus, the whole population size is adapted during the run by the competition model.

Numerical results for difficult test functions can be found in [45]. A discussion about he evolution of the population sizes during a run can be found in [44].

8 Conclusion

Complex spatial population structures are seldom used in evolutionary computation. In this chapter we have investigated the stepping-stone model, competing sub-populations, and Darwin's continent-island cycle. For Darwin's conjecture an evolutionary algorithm was used where the fitness of each individual is given by the competition with other individuals. The competition is modeled by evolutionary games. The parallel genetic algorithm PGA uses the stepping-stone interaction. It runs totally in parallel. The selection is distributed and done by each individual in its neighborhood. Faster convergence can be obtained by the Breeder Genetic Algorithm BGA. It models breeding as it is done by a human breeder. For really difficult optimization problems the competing BGA has been developed. It uses competing sub-populations which are bred using different strategies. Occasionally good individuals migrate to other sub-populations. The sizes of the sub-populations are adjusted according to their performance.

Darwin's cycle model seems also a good starting point for investigating the development of new ideas in human societies, be it in science or art. It takes small groups or even a single individual to try out new ideas. But for the ideas to be accepted a large community is needed. In a large community many individuals evaluate the new ideas, only the most promising eventually survive.

References

1. Axelrod, R.: The evolution of cooperation. Basic, New York (1984)
2. Axelrod, R.: The evolution of strategies in the iterated prisoner's dilemma. In: Davis, L. (ed.) Genetic algorithms and Simulated Annealing, pp. 32–41. Morgan Kaufmann, Los Altos (1987)
3. Batz, M.: Evolution von Strategien des Iterierten Gefangenen Dilemma. Master's thesis, Universität Bonn (1991)
4. Cavalli-Sforza, L.L., Feldman, M.W.: Cultural Transmission and Evolution: A Quantitative Approach. Princeton University Press, Princeton (1981)
5. Cohoon, J.P., Hedge, S.U., Martin, W.N., Richards, D.: Punctuated equilibria: A parallel genetic algorithm. In: Grefenstette, J.J. (ed.) Proceedings of the Second International Conference on Genetic Algorithms, pp. 148–154. Lawrence Erlbaum, Mahwah (1987)
6. Crow, J.F., Engels, W.R., Denniston, C.: Phase three of wright's shifting balance theory. Evolution 44, 233–247 (1990)
7. Dawkins, R.: The Selfish Gene, 2nd edn. Oxford University Press, Oxford (1989)
8. Emlen, J.M.: Population Biology: The Coevolution of Population Dynamics and Behavior. Macmillan Publishing Company, New York (1984)
9. Felsenstein, J.: A pain in the torus: Some difficulties with models of isolation by distance. Amer. Natur. 109, 359–368 (1975)
10. Felsenstein, J.: The theoretical population genetics of variable selection and migration. Ann. Rev. Genet. 10, 253–280 (1976)
11. Fisher, R.A.: The Genetical Theory of Natural Selection. Dover, New York (1958)
12. Glover, F.: Heuristics for integer programming using surrogate constraints. Decision Sciences 8, 156–166 (1977)
13. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
14. Gorges-Schleuter, M.: Asparagos: An asynchronous parallel genetic optimization strategy. In: Schaffer, H. (ed.) 3rd Int. Conf. on Genetic Algorithms, pp. 422–427. Morgan-Kaufmann, San Francisco (1989)
15. Gorges-Schleuter, M.: Genetic Algorithms and Population Structures - A Massively Parallel Algorithm. PhD thesis, University of Dortmund (1991)
16. Gould, S.J., Eldredge, N.: Punctuated equilibria: the tempo and mode of evolution reconsidered. Paleobiology 3, 115–151 (1977)
17. Griffiths, P.E.: The philosophy of molecular and developmental biology. In: Blackwell Guide to Philosophy of Science. Blackwell Publishers, Malden (2002)
18. Gurney, W.S.C., Nisbet, R.M.: Ecological Dynamics. Oxford University Press, New York (1998)
19. Haeckel, E.: Über die Entwicklungstheorie Darwin's. In: Gemeinverständliche Vorträge und Abhandlungen aus dem Gebiet der Entwicklungslehre. Emil Strauss, Bonn (1902)
20. Hamilton, W.D.: The genetical evolution of social behavior I and II. Journal of Theoretical Biology 7, 1–16, 17–52 (1964)
21. Hofbauer, J., Sigmund, K.: Evolutionary Games and Population Dynamics. Cambridge University Press, Cambridge (1998)
22. Holland, J.H.: Adaptation in Natural and Artificial Systems. Univ. of Michigan Press, Ann Arbor (1975/1992)
23. Jablonka, E., Lamb, M.J.: Evolution in Four Dimensions. MIT Press, Cambridge (2005)
24. Johnson, D.S.: Local optimization and the traveling salesman problem. In: Paterson, M.S. (ed.) Automata, Languages and Programming. LNCS, vol. 496, pp. 446–461. Springer, Heidelberg (1990)

25. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. Operations Research 37, 865–892 (1989)
26. Johnson, D.S., McGeoch, L.A.: The traveling salesman problem: a case study. In: Aarts, E., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 215–310. Wiley, Chichester (1997)
27. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal 2, 291–307 (1970)
28. Lin, S.: Computer solutions of the traveling salesman problem. Bell. Syst. Techn. Journ. 44, 2245–2269 (1965)
29. Mahnig, T., Mühlenbein, H.: A new adaptive Boltzmann selection schedule SDS. In: Proceedings of the 2001 Congress on Evolutionary Computation, pp. 183–190. IEEE Press, Los Alamitos (2001)
30. Manderick, B., Spiessens, P.: Fine-grained parallel genetic algorithm. In: Schaffer, H. (ed.) 3rd Int. Conf. on Genetic Algorithms, pp. 428–433. Morgan-Kaufmann, San Francisco (1989)
31. Marks, R.E.: Breeding hybrid strategies: Optimal behavior for oligopolist. In: Schaffer, H. (ed.) 3rd Int. Conf. on Genetic Algorithms, pp. 198–207. Morgan Kaufmann, San Mateo (1989)
32. Maturana, H.R., Varela, F.J.: Autopoiesis and Cognition: The Realization of the Living. D. Reidel, Boston (1980)
33. Miller, J.K.: The coevolution of automata in the repeated prisoner's dilemma. Technical report, Santa Fe Institute (1989)
34. Mühlenbein, H.: Evolution in time and space - the parallel genetic algorithm. In: Rawlins, G. (ed.) Foundations of Genetic Algorithms, pp. 316–337. Morgan Kaufmann, San Mateo (1991)
35. Mühlenbein, H., Gorges-Schleuter, M., Krämer, O.: Evolution algorithms in combinatorial optimization. Parallel Computing 7, 65–88 (1988)
36. Mühlenbein, H., Höns, R.: The factorized distribution algorithm and the minimum relative entropy proncíple. In: Pelikan, M., Sastry, K., Cantu-Paz, E. (eds.) Scalable Optimization via Probabilistic Modeling, pp. 11–37. Springer, New York (2006)
37. Mühlenbein, H., Mahnig, T.: Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. Journal of Approximate Reasoning 31(3), 157–192 (2002)
38. Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization. Evolutionary Computation 1, 25–49 (1993)
39. Mühlenbein, H., Schlierkamp-Voosen, D.: The science of breeding and its application to the breeder genetic algorithm. Evolutionary Computation 1, 335–360 (1994)
40. Oyama, S.: Evolutions's Eye. Duke University Press, Durham (2000)
41. Padberg, W., Rinaldi, G.: Optimization of a 532-city symmetric traveling saleman problem by branch and cut. Op. Res. Let. 6, 1–7 (1987)
42. Parisi, D., Ugolini, M.: Living in enclaves. Complexity 7, 21–27 (2002)
43. Rapaport, A.: Modern systems theory – an outlook for coping with change. General Systems XV, 15–25 (1970)
44. Schlierkamp-Voosen, D., Mühlenbein, H.: Strategy adaptation by competing subpopulations. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) PPSN 1994. LNCS, vol. 866, pp. 199–208. Springer, Heidelberg (1994)
45. Schlierkamp-Voosen, D., Mühlenbein, H.: Adaptation of population sizes by competing subpopulations. In: Proceedings IEEE Conference on Evolutionary Computation, pp. 330–335. IEEE Press, New York (1996)

46. Maynard Smith, J.: Evolution and the Theory of Games. Cambridge University Press, Cambridge (1982)
47. Maynard Smith, J., Szathmary, E.: The Major Transitions in Evolution. W.H. Freeman, Oxford (1995)
48. Tanese, R.: Distributed genetic algorithm. In: Schaffer, H. (ed.) 3rd Int. Conf. on Genetic Algorithms, pp. 434–440. Morgan-Kaufmann, San Francisco (1989)
49. Ulder, N.L.J., Pesch, E., van Laarhoven, P.J.M., Bandelt, H.-J., Aarts, E.H.L.: Improving tsp exchange heuristics by population genetics. In: Maenner, R., Schwefel, H.-P. (eds.) Parallel Problem Solving from Nature, pp. 109–116. Springer, Heidelberg (1991)
50. Voigt, H.-M., Mühlenbein, H.: Gene Pool Recombination and the Utilization of Covariances for the Breeder Genetic Algorithm. In: Michalewicz, Z. (ed.) Proc. of the 2nd IEEE International Conference on Evolutionary Computation, pp. 172–177. IEEE Press, New York (1995)
51. Voigt, H.-M., Mühlenbein, H., Cvetković, D.: Fuzzy recombination for the continuous breeder genetic algorithm. In: Eshelman, L.J. (ed.) Proc. of the Sixth Int. Conf. on Genetic Algorithms, pp. 104–112. Morgan Kaufmann, San Francisco (1995)
52. von Laszewski, G.: Ein paralleler genetischer Algorithmus für das Graph Partitionierungsproblem. Master’s thesis, Universität Bonn (1990)
53. von Laszewski, G.: Intelligent structural operators for the k-way graph partitioning problem. In: Belew, R.K., Booker, L. (eds.) Procedings of the Fourth International Conference on Genetic Algorithms, pp. 45–52. Morgan Kaufmann, San Mateo (1991)
54. Wilson, D.S., Dugatkin, L.A.: Nepotism vs tit-for-tat, or, why should you be nice to your rotten brother. Evol. Ecology 5, 291–299 (1991)
55. Wright, S.: The distribution of gene frequencies in populations. Proc. Nat. Acad. Sci 24, 253–259 (1937)
56. Wright, S.: Factor interaction and linkage in evolution. Proc. Roy. Soc. Lond. B 162, 80–104 (1965)

Part IX
CI for Clustering and Classification

Fuzzy Clustering of Likelihood Curves for Finding Interesting Patterns in Expression Profiles

Claudia Hundertmark, Lothar Jänsch, and Frank Klawonn

Abstract. Peptides derived from proteins are routinely analysed in so-called bottom-up proteome studies to determine the amounts of corresponding proteins. Such studies easily sequence and analyse thousands of peptides per hour by the combination of liquid chromatography and mass spectrometry instruments (LC-MS). However, quantified peptides belonging to the same protein do not necessarily exhibit the same regulatory information in all cases. Several causes can produce these regulatory inconsistencies at the peptide level. Quantitative data might be simply influenced by specific properties of the analytical procedure. However, it can also indicate meaningful biological processes such as the post-translational modification (PTM) of amino acids regulated in individual protein regions. This article describes a fuzzy clustering approach allowing the automatic detection of regulatory peptide clusters within individual proteins. The approach utilises likelihood curves to summarise the regulatory information of each peptide, based on a noise model of the used analytical workflow. The shape of these curves directly correlates with both the regulatory information and the underlying data quality, serving as a representative starting point for fuzzy clustering of peptide data assigned to one protein.

1 Introduction

Cellular processes are mediated by proteins acting e.g. as enzymes in different metabolic or signalling pathways. Their activity is determined by (i) their abundance

Claudia Hundertmark
Department for Cell Biology, Helmholtz Centre for Infection Research,
Inhoffenstr. 7, D-38124 Braunschweig, Germany
e-mail: claudia.hundertmark@helmholtz-hzi.de

Lothar Jänsch
Department for Cell Biology, Helmholtz Centre for Infection Research,
Inhoffenstr. 7, D-38124 Braunschweig, Germany
e-mail: lothar.jaensch@helmholtz-hzi.de

Frank Klawonn
Department of Computer Science, University of Applied Sciences,
Braunschweig/Wolfenbuettel, Salzdahlumer Str. 46/48, D-38302 Wolfenbuettel, Germany
e-mail: f.klawonn@fh-wolfenbuettel.de

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 599–622.
springerlink.com

controlled by gene expression, and (ii) modifications made following (post) their synthesis (translation) at ribosomes. These post-translational modifications (PTMs) alter the chemical structure of protein-constituting amino acids. The modifications occur only at specific regions of the protein sequence and often control essential intra- and intermolecular binding and activity properties of the modified proteins. Therefore, proteome research, i.e. the systematic characterisation of proteins, aims to develop quantitative strategies suitable for both protein expression and PTM analyses. In so-called bottom-up approaches proteins are routinely digested first into peptides, resulting in complex samples comprising unmodified and modified protein regions. Following this, all peptides are separated by liquid chromatography and can be analysed quantitatively by mass spectrometry (LC-MS). Thus, a comparative investigation of peptides derived from cells in different physiological states or/and under variable environmental conditions provides data that characterizes proteins as well as the post-translational modifications that are involved in biological processes.

If a protein has changed its function at the expression level, it is likely that peptides representing different regions of this protein will be regulated in a consistent manner. In contrast, scientists have to consider PTMs if individual peptides presenting individual protein regions are differently regulated.

Regulatory information can simply be presented as ratios (regulation factor, expression ratio), which are calculated from pairwise comparisons of detected amounts of the same peptide under different conditions. However, calculating the expression states of proteins itself requires a statistical strategy to combine the regulatory information of peptides belonging to the same protein correctly. Signal intensities in mass spectrometry are compromised by noise, resulting in variable expression ratios even under assumed constant experimental conditions. Thus, regulatory peptide data for one protein are often similar but never identical, raising the question: which variation is caused by simple noise and which indicates individually regulated protein regions that should be excluded from general protein expression calculations?

We have recently established a noise model-based workflow for the iTRAQ$^{\text{TM}}$ technology frequently used in quantitative proteome research [8]: Different types of iTRAQ$^{\text{TM}}$ reporter molecules are linked to peptides and produce sample-specific ions (reporter masses of 114, 115, 116 or 117 Dalton) during MS analyses. The measured relative reporter intensities correlate with the relative ratios of peptides in comparatively analysed biological samples. Following this, a mathematical model calculates the noise inherent in a peptide's regulatory information that is generally decreasing with increasing intensity of the detected iTRAQ$^{\text{TM}}$ reporter signals. The noise model allows calculation of both the most likely regulation factor and the probability of alternative regulations, based on the underlying MS data qualities. Both aspects are graphically summarised in likelihood curves that were established systematically for all regulatory peptide data. Overlapping curves of peptides belonging to the same protein often form a kind of main cluster indicating the general expression state of the total protein. In contrast, the regulation of individual protein regions is probably significant, if the likelihood curve of the corresponding peptide does not substantially overlap with other curves of the main cluster. Those outlying curves are called *outliers*. Importantly, these cases can detect regulated PTMs but also can reveal peptides that cannot be assigned unambiguously to one protein.

Therefore, computer-aided clustering of regulatory peptide data is a feasible resource for both quality management in proteome studies and the detection of important biological processes, such as post-translational modifications. In this chapter a prototype-based fuzzy clustering approach is presented to inspect the variations of regulatory peptide data at the protein level. In a first step, regulatory information is calculated and visualised by a probabilistic approach resulting in likelihood curves for each individual peptide. Then, the likelihood calculations for all peptides belonging to one protein are inspected by fuzzy clustering in order to detect outlying curves. Since the algorithm for the detection of peptide clusters is based on fuzzy clustering, our collaborative approach combines probabilistic concepts as well as principles from soft computing. However, fuzzy clustering is usually based on data points and its application to likelihood curves was a challenging task. An integrative concept is presented and discussed in this article with particular respect to distance and quality measures.

2 Background to Quantitative Proteomics and iTRAQ$^{\text{TM}}$ Based Likelihood Curves

Important classes of cellular bio-molecules (e.g. proteins) have become accessible by way of different high throughput technologies, and can be analysed systematically to define and discover the molecular basis of life. Active genes are transcribed into messenger RNA (mRNA) which is translated into proteins by ribosomes. Besides structural properties, the majority of proteins work as enzymes mediating nearly every type of cellular function by controlling the metabolism and signalling networks. Chemical products resulting from enzyme activities are generally termed metabolites.

From the biological perspective, it is very important to compare measurable properties for the same item (mRNA, proteins, metabolites) under different conditions and to conclude unambiguously (i) whether they differ significantly and (ii) to reveal the level of regulation. Typically, tested conditions focus on the consequences of, e.g. variable environmental or physiological settings (treated vs. untreated; aerobe vs. anaerobe), a different genetic background (wild type vs. mutant) or molecular processes in human diseases (normal/healthy cell vs. infected/cancerogenic cell).

mRNA levels can be investigated on micro array-based imaging systems detecting fluorescence signals, and these strategies are termed transcriptomics. Accordingly, the systematic investigation of proteins and metabolites was named proteomics and metabolomics, respectively. Both research directions depend decisively on mass spectrometry (MS) as a read out system for their quantification. In addition to two-dimensional polyacrylamide gel electrophoresis (2D-PAGE), proteins with highly diverse biochemical properties are nowadays routinely investigated by the combination of liquid chromatography and MS, allowing automated amino acid sequencing (LC-MS/MS). Similar, metabolite concentrations are determined by combining gas chromatography and MS (GC-MS).

2.1 Proteomics

Protein-dependent gene regulation determines the selection and synthesis of different mRNAs, resulting in the translation of proteins at ribosomes. Proteins are large organic compounds made of 20 different types of amino acid and consist of up to several thousands of amino acids. Short sequences comprising less then 100 amino acids are usually termed peptides.

Once produced, proteins mediate, control and regulate almost all cellular processes and establish the physiological and reactive capacities of organisms. Some proteins have structural or mechanical functions, such as actin in the cytoskeleton or myosin in muscles. However, the majority of proteins act as enzymes, which catalyse chemical reactions. Each organic compound (e.g. metabolite) is the product of enzymatic activities. In addition, proteins constitute a dynamic network which transmits and integrates environmental and internal signals that are indispensable for cellular communications. The molecular interactions of signalling proteins are frequently regulated by post-translational modifications which are again catalysed by enzymes. Currently, about 200 [13] different modifications have been described for proteins, altering their structure and concomitantly their activity state, localisation or stability.

In contrast to the genome, the proteome, i.e. the sum of all proteins of a cell is *per se* highly dynamic and varies significantly with regard to its qualitative and quantitative composition during the cell cycle and depending on the environmental conditions. Proteomics aims at the identification and representative characterisation of all proteins in a cell under defined conditions. Since technologies for absolute protein quantifications are still limited and the corresponding strategies very time and cost intensive, proteome studies are usually comparative, yielding relative quantifications. The quantifications are based either on staining procedures and resulting signals from gel separated proteins or in the case of LC-MS/MS based on labelling strategies and sample specific ion intensities (see below).

2.2 Identification and Characterisation of Proteins Based on LC-MS

Mass spectrometry (MS) is an analytical technique that measures the mass-to-charge ratio of charged, i.e. ionised particles. The ionisation can be achieved by laser energy (Maldi Assisted Laser Desorption Ionisation, MALDI) or by electrospray ionisation (ESI) of appropriate solvents containing the analytes. Characteristic masses can be obtained from total proteins, from protein derived peptides or fragments comprising partial amino acid sequences. Amino acid-specific ion-fragmentation patterns are usually generated in so-called tandem mass spectrometry experiments (MS/MS) and allow the automated sequencing of peptides and the site-specific detection of post-translational modifications. Since proteomes are often highly complex samples comprising thousands of proteins, further biochemical or chromatographic separation strategies have to be combined with MS to achieve representative analyses.

2D-PAGE was considered the "work horse" in comparative proteome studies for many years. Separated protein spots and corresponding intensities can be analysed by specific software packages defining regulated proteins and often providing statistical information about the robustness of this analytical procedure. Regulated protein spots are usually identified individually based on MALDI-MS and a method called "mass fingerprinting". Actually, 2D-PAGE is still the method of choice for investigating complex proteomes at the level of intact proteins. However, 2D-PAGE has only limited capabilities for the characterisation of hydrophobic proteins or those that exhibit extreme isoelectric point (pI) or molecular mass (MW) values. In contrast, the combination of liquid chromatography directly linked with MS (LC-MS) is to a large extent compatible with proteins of highly diverse biochemical properties. In bottom-up proteomics, proteins are digested into peptides that are separated by liquid chromatography methods. The last LC preferentially elutes only a small number of peptides at a particular time into a mass spectrometer where they become ionised (ESI) and are characterised with respect to their mass to charge ratios (m/z). Instruments used in proteomics usually provide additional tandem MS functionality, i.e. peptide ions are selected individually for fragmentation experiments. Fragmentations occur "randomly" at different positions in the peptide backbone of the amino acid sequence. Therefore, the mass increments of fragments correspond to the order of neighbouring amino acids in the peptide and can be utilised for the systematic sequencing as well as for the detection of modified amino acids. Expected fragment ions can also be calculated *in silico* based on the sequence information stored in protein databases. The comparison of *in silico* and experimentally generated fragmentation patterns is utilised in best fit approaches for the high throughput sequencing of peptides in automated LC-MS/MS workflows. Thus, LC-MS/MS basically allows sequencing and unambiguous identification of hundreds to thousands of peptides per hour, yet only provides limited information about the abundance of the investigated components. Several physical and biochemical parameters contribute to the process of peptide ionisation, hampering reliable quantifications at this time because the calculation of peptide and protein abundances is based on corresponding ion intensities and does not provide the data quality that can be achieved by using labels. In order to overcome this limitation in LC-MS, several labelling strategies were established for quantitative proteomics. The key benefit of these approaches is the possibility of combining differentially labelled peptides before LC-MS analyses. Consequently, peptides with identical sequence from different samples are equally affected by parameters varying the ionisation efficiency during the analytical process.

2.3 Quantification of Peptides and Proteins

Besides identification, mass spectrometry is used for the relative quantification of proteins. Comparative analyses are performed in order to reveal proteins regulated under specific conditions and to define the networks, processes and signalling pathways involved. LC-MS/MS typically investigates protein-derived peptides. Thus, strategies for the quantification and characterisation of proteins should preferentially assess the peptide level.

Besides SILAC [10], iTRAQ^{TM} – introduced by [12] in 2004 – became the standard for relative quantifications of automatically sequenced peptides. iTRAQ^{TM} allows differential labelling and relative as well as absolute peptide quantification of up to eight different samples in parallel. During the labelling process only one type out of eight iTRAQ^{TM} molecules is linked covalently to every peptide from one biological sample. All eight iTRAQ^{TM} molecules have the same structure and molecular weight, but differ in the distribution of incorporated isotopes (Figure 1). In the intact molecule the total mass is balanced and each labelling reaction introduces an identical mass shift. However, under the conditions of peptide sequencing (MS/MS) iTRAQ^{TM} also produces fragment ions that differ in mass and serve as sample specific reporters: Same peptides (with identical amino acid sequence) from different biological samples, which were labelled differentially and are subsequently pooled exhibit the same biochemical properties and total masses. Consequently, identical peptides from different samples co-elute at the same time from chromatographic columns, enter with the same molecular weight the MS device and are subjected commonly to the fragmentation process. The ratios of the released iTRAQ^{TM} reporter ions correlate with the relative abundance of the analysed peptides as part of the investigated samples.

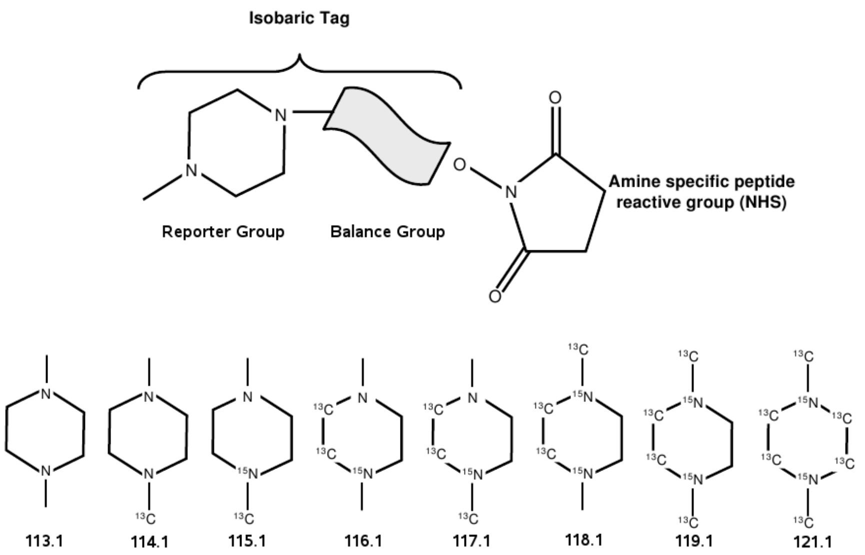


Fig. 1 Chemical constitution of the iTRAQ^{TM} molecules: reporter group, balance group and reactive group (taken from [11])

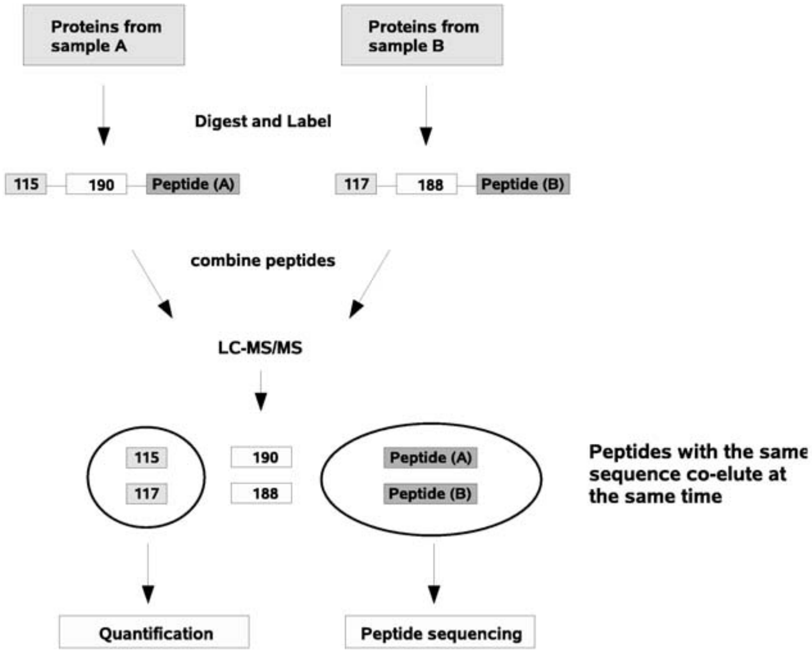


Fig. 2 iTRAQ^{TM} workflow: proteins from samples A and B are digested, peptides are iTRAQ^{TM} labelled and combined. When performing LC-MS/MS peptide bonds between the amino acids as well as bonds of iTRAQ^{TM} molecules and peptides are broken. Subsequently, peptides are used for identification and iTRAQ^{TM} molecules are used for relative quantification

A typical iTRAQ^{TM} workflow, allowing the relative quantification of peptides derived from proteins of two different samples, is summarised in Figure 2. Initially, all proteins from both samples are cleaved into peptides by a specific endo-protease (e.g. digestion with Trypsin). Thereafter, both peptide fractions are labelled separately using different iTRAQ^{TM} reagents, each containing reporter groups of different masses (e.g. 115 Dalton or 117 Dalton of mass). iTRAQ^{TM} molecules are linked covalently to the N-terminus of each peptide as well as to every present lysine in the peptide sequence. Identical peptides from different samples exhibit a modified but identical chemical behaviour and mass subsequent to the iTRAQ^{TM} labelling. Therefore, differentially labelled samples can be pooled and subjected to LC to decrease the sample complexity before the MS analyses. Whereas the amino acids of peptides from both samples commonly contribute to the total ion intensities used for the peptide sequencing, the reporters dissociate in sample-specific amounts. Reporter ions with 115 Da and 117 Da can be detected as part of every peptide fragmentation spectrum and a direct comparison of their intensities gives information about the relative abundance of peptides in the compared samples.

Table 1 Two different peptides are found in both analysed samples. The samples were labelled quantitatively with iTRAQ^{TM} reagents 115 and 117 that were experimentally found with ion intensities of 200 and 400, respectively. Therefore, the 115-labelled sample contains half the amount of peptide 1 than the 117-labelled sample (regulation factor = 0.5). In constrast, the fragmentation pattern of peptide 2 exhibited reporter intensities of 80 and 40 indicating a regulation factor of 2

Peptide	intensity iTRAQ^{TM} 115	intensity iTRAQ^{TM} 117	regulation factor
Peptide 1	200	400	0.5
Peptide 2	80	40	2

Regulatory information can be presented as ratios (regulation factor, expression ratio), which are calculated from pairwise comparisons of iTRAQ^{TM} reporter ions of peptides from all samples. Table 1 gives an example for an experiment with two peptides found in two samples, which were labelled with iTRAQ^{TM} 115 and iTRAQ^{TM} 117.

2.4 Impreciseness of Regulatory Information: Intensity-Dependent Noise

The quality of measured intensities is not the same for all detectable intensity values: in comparison, accuracy of high intensities is significantly better than accuracy of low intensities. This effect has been observed previously [6, 9] and can be demonstrated by calculating and plotting the regulation factors (intensity ratios) of two equally regulated samples (Figure 3). The simulation of such a case can be achieved by differentially labelling and combining two parts from the same original sample. According to the presented workflow in Figure 2, both aliquots are digested, labelled separately with different iTRAQ^{TM} reagents and then combined (test dataset). Following this, intensity ratios of all detected peptides in both compared samples are calculated (cf. Table 1). The result of two samples prepared that way and analysed by LC-MS/MS is given in Figure 3, and regulation factors are plotted vs. intensity of the iTRAQ^{TM} 115 labelled sample. For the sake of clarity, intensities are logarithmic transformed (base two). The horizontal line gives the position where the intensity ratio was placed in case of no regulation of the compared samples. Intensity ratios on the left hand side of the plot derived from low intensities are deviating considerably from the line compared to the ratios derived from high intensities on the right. Hence, there is no doubt that ratios derived from high intensities are significantly more accurate than ratios derived from low intensities. In the following, we refer to this effect as *intensity-dependent noise*.

2.4.1 Noise Model

The high fluctuation of expression ratios, particularly at low intensities, originates from the imprecision of measured intensities. Small errors of measurement affect expression ratio calculation more while detecting low intensities than it is the case with high intensities. Therefore, the noise inherent in intensity measurements must be estimated. How can this be done?

Each intensity is measured several times. In the ideal case without noise and without regulation, all measurements should be identical. Since noise cannot be ruled out, it is impossible to know the true intensities. Due to the fact that normally only very few samples (between two and eight) are analysed in parallel, available data do not conform to statistical methods. Hence, performing noise estimation from the sample to be analysed is not recommended. In order to specify the noise characteristics, we prepared a special training dataset. We repeatedly analysed selected synthesised and iTRAQ$^{\mathrm{TM}}$ labelled peptides, generating intensities over a broad dynamic range. Of course, the obtained data are only reliable specifically for that instrument, which was used for the measurements.

The noise follows a log-normal distribution whose variance depends on the (true) intensity. It does not seem appropriate to assume a normal distribution of the noise directly, since intensities are always non-negative. Since calculations are much easier with normal distributions, in most cases we will consider the data after taking their logarithm.

The general problem to be solved is as follows. A data set of the following form is given: $\left(y_1^{(1)},\ldots,y_1^{(k_1)},\ldots,y_n^{(1)},\ldots,y_n^{(k_n)}\right)$. (Here we use the transformed data.)

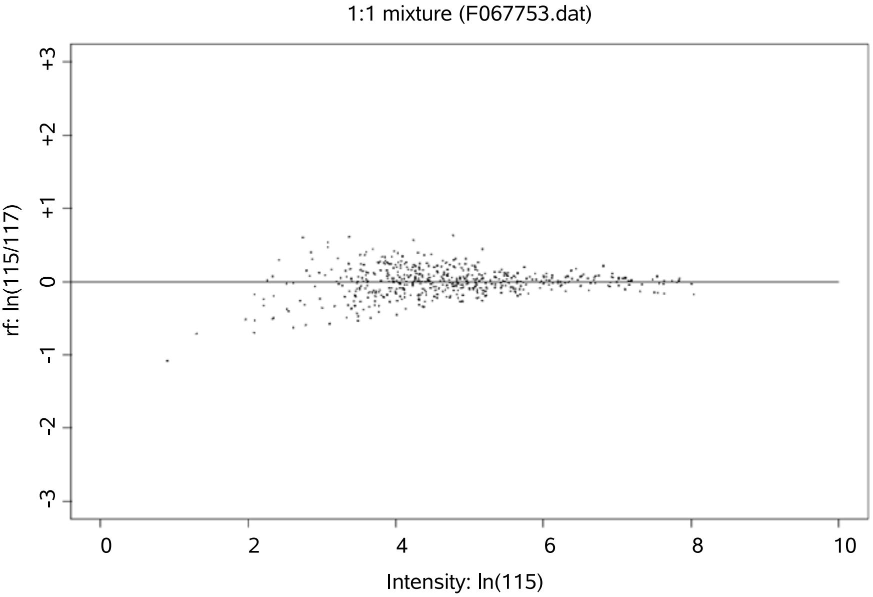


Fig. 3 Regulation factors versus intensity of iTRAQ$^{\mathrm{TM}}$ 115 labelled sample (both logarithmic transformed). Intensity-dependent noise: ratios derived from high intensities are significantly more accurate than ratios derived from low intensities

$y_i^{(1)},\dots,y_i^{(k_i)}$ represents k_i noisy measurements of the same (logarithmic) unknown intensity μ_i.

We assume that the subsample $y_i^{(1)},\dots,y_i^{(k_i)}$ originates from independent samples of a normal distribution with unknown mean μ_i and unknown variance σ_i. From experiments we know that the variances follow a certain tendency. Small intensities are less reliable (more noisy) than larger ones. In order to take this into account, we assume that we have

$$\sigma(\mu_i) \;=\; a + re^{-\lambda\mu_i} \tag{1}$$

with $a, r, \lambda \geq 0$. a represents the absolute noise in the measurement that is always present. r specifies the amount of relative noise depending on the (logarithmic) intensity μ_i. λ determines how fast the relative noise decreases with increasing (logarithmic) intensity. The aim is to estimate the parameters a, r and λ based on the sample $\left(y_1^{(1)},\dots,y_1^{(k_1)},\dots,y_n^{(1)},\dots,y_n^{(k_n)}\right)$. Unfortunately, this requires that we estimate μ_i for each subsample $y_i^{(1)},\dots,y_i^{(k_i)}$. We only know that the values in the subsample are noisy measurements of the same intensity μ_i, but we do not know μ_i. We carry out a maximum likelihood estimation based on an EM (expectation maximisation) strategy. By estimating the parameters a, r and λ, we want to maximise the likelihood

$$L\left(y_1^{(1)},\dots,y_1^{(k_1)},\dots,y_n^{(1)},\dots,y_n^{(k_n)}|a,r,\lambda\right) \;=$$

$$\prod_{i=1}^{n}\prod_{j=1}^{k_i}\frac{1}{(a+re^{-\lambda\mu_i})\sqrt{2\pi}}\exp\left(-\frac{(y_i^{(j)}-\mu_i)^2}{2(a+re^{-\lambda\mu_i})^2}\right). \tag{2}$$

The factors are simply the densities of normal distributions with mean μ_i and deviation $\sigma_i = a + re^{-\lambda\mu_i}$.

As mentioned before, the maximisation of L does not only involve the determination of the parameters a, r and λ, but also the estimation of the μ_i. Assuming the parameters a, r and λ to be fixed at the moment, we estimate the μ_i-values in the following way. Since the maximisation of the log-likelihood is equivalent to the maximisation of the likelihood itself, we consider – as usual in maximum likelihood estimation – the log-likelihood. When the parameters a, r and λ are fixed, the μ_i-values can be optimised independently. This means we have to maximise the log-likelihoods

$$\tilde{L}_i \;=\; \sum_{j=1}^{k_i}\left(-\ln(\sqrt{2\pi}) - \ln(h(\mu_i)) - \frac{(y_i^{(j)}-\mu_i)^2}{2(h(\mu_i))^2}\right) \tag{3}$$

where $h(\mu_i) = a + re^{-\lambda\mu_i}$. In order to maximise $\tilde{L}_i$ it is necessary that

$$\frac{d\tilde{L}_i}{d\mu_i} = \sum_{j=1}^{k_i} \Bigg(-\frac{h'(\mu_i;\theta)}{h(\mu_i;\theta)} + \frac{(x_i^{(j)}-\mu_i)(h(\mu_i;\theta)+(x_i^{(j)}-\mu_i)h'(\mu_i;\theta))}{h^3(\mu_i;\theta)} \Bigg) = 0 \tag{4}$$

holds. With $h'(\mu_i) = -\lambda r e^{-\lambda\mu_i}$ and multiplying (4) by $(a+re^{-\lambda\mu_i})^3$ we obtain

$$\frac{d\tilde{L}_i}{d\mu_i} = \sum_{j=1}^{k_i} \Bigg((\lambda r e^{-\lambda\mu_i})(a+re^{-\lambda\mu_i})^2 + (x_i^{(j)}-\mu_i)((a+re^{\lambda\mu_i}) + (x_i^{(j)}-\mu_i)(-\lambda re^{-\lambda\mu}) \Bigg) = 0 \tag{5}$$

Solving (5) for μ_i yields the maximum likelihood estimation for μ_i, assuming the parameters a, r and λ to be fixed. This is done in a numerical manner by a simple bisection strategy. As one boundary for bisection, we choose the mean value of the $y_i^{(j)}$. The second one is determined by systematically searching to the left and right from this value until the sign of (5) changes.

The optimisation of the parameters a, r and λ is carried out by a stochastic heuristic algorithm, an evolution strategy [1] with adaptive mutation rates, population size = 10, number of children = 25, maximum tolerated number of succeeding generations in which no improvement could be achieved = 20 and maximum number of iterations = 200. The fitness of a parameter combination (a, r, λ) is given by (2), where the μ_i are determined as described above based on solving (5).

2.5 Calculation and Visualisation of Regulatory Information

In Section 2.3 we introduced an intuitive method for the calculation of regulation factors at the level of peptides. Indeed, this idea does not consider the different quality of intensities presented in the previous section. Furthermore, we have to establish a concept for the calculation of representative protein regulation factors based on individual peptide information. Besides, sometimes in MS/MS single peptides are detected several times (possibly varying in length) with slightly differing regulation or several MS/MS results are merged. The outcome is a dataset containing identical peptides several times. All these cases require the calculation of the most suitable regulation factor for the total group of peptides as well.

As an application of our noise model we developed an intuitive concept for the visualisation-aided exploration of regulatory iTRAQ$^{\mathrm{TM}}$ data based on likelihood curves precisely depicting the overall data-dependent quality of regulatory information [7]. This approach can be applied in a cumulative way at both the protein and the peptide level.

All of the likelihood plots that are presented in the following examples are derived from proteins of differentially stimulated and quantitatively analysed cells.

2.5.1 Calculation of Regulatory Information

An approach to the calculation of the best fitting regulation factor (expression ratio) for a group of peptides was derived from the noise model. Generally, this method can be applied to two different cases; firstly, the calculation of protein regulation factors. Several proposals were submitted in the last few years [4, 9], but until now none of those considered the reliability of the underlying peptide signals. The second application is the calculation of expression ratios of multiple observed identical peptides. As usual in MS/MS, the identification of every detected peptide is linked to an MS/MS spectrum. Since multiple selection for fragmentation of one peptide is not unusual, MS/MS datasets may contain redundancies in the form of multiple measured peptides. In the following we will refer to every single identification of a peptide as a *matched MS/MS spectrum.*

According to the noise model, expression ratios derived from high intensities have to outweigh the expression ratios derived from low intensities. In order to find the most suitable expression ratio for a peptide from a group of possibly different expression ratios we scan all expression ratios c_j inside an individually specified interval $[c_{min},\ldots,c_{max}]$ for all MS/MS spectra that match the actual considered peptide. A likelihood value l_j is determined corresponding to every c_j. The most suitable overall expression ratio c_{best} is that one which results in the maximum likelihood value l_{best}.

In addition to the calculation of a protein's and peptide's best fitting regulation factor, this method can be used for the detection of mismatched peptides. Often, parts of related proteins are identical in their amino acid sequence. A peptide that was identified by MS/MS can not be matched unambiguously to one of the related proteins if its sequence is part of more than one protein. Thus, for regulated peptides it has to be tested whether their sequence occurs exclusively only in one protein. If not, such regulatory events can indicate to significant mistakes in the identification reports. The determination of outliers is done by fuzzy clustering (Section 4).

2.5.2 Visualisation of Regulatory Information

All of the likelihood values l_j and the corresponding expression ratios c_j are plotted after area normalising $\int_{c_{min}}^{c_{max}} = 1$ to enhance the clarity of visualisation. Robustness of the underlying data is proportional to both the height and the slope of the produced curve .

Depending on the aim of analysis, various views of a protein may be useful. Peptides can be combined and visualised by a shared curve or separately by individual curves within the plot. Figure 4 shows all peptides of a protein given by a single likelihood curve representing the total protein. In the next figure all different peptides are presented separately by individual curves (Figure 5). Finally, both perspectives

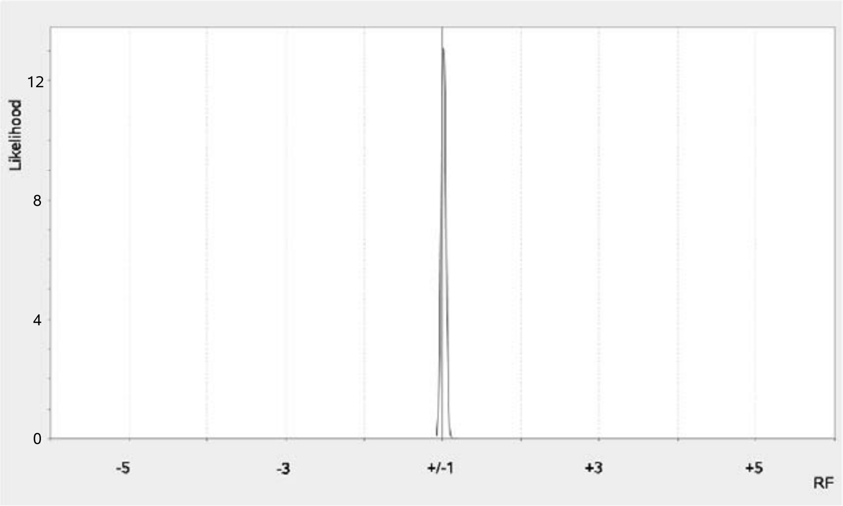


Fig. 4 Protein CSK21_HUMAN: 5 peptides combined to a single likelihood curve representing the total protein

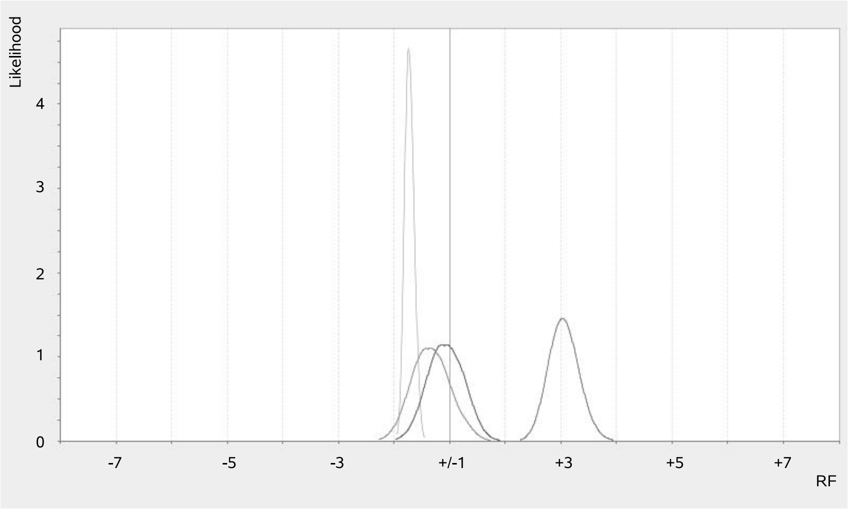


Fig. 5 Protein NEK9_HUMAN: 3 peptides slightly down-regulated, 1 peptide up-regulated by factor 3

can be combined in such a way that some special peptides can be plotted separately in contrast to the remaining peptides of the protein, which are given by the protein curve (Figure 6).

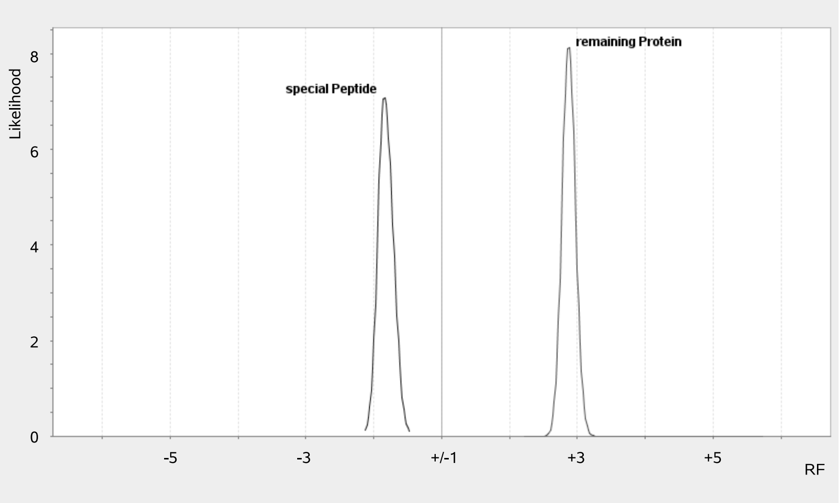


Fig. 6 Protein K1C14_HUMAN: 3 peptides combined to the protein likelihood curve, 1 peptide represented by an individual curve

3 Fuzzy Cluster Analysis

Discovering unusual deviations in regulatory behaviour is the main focus of the investigation of the likelihood curves in this paper. In the ideal case, the likelihood curves for the regulation of peptides from the same protein should be more or less similar. However, due to the different intensity of the noise, depending on the measured intensities for the peptides, the likelihood curves might be narrower in case of low noise and broader in case of higher noise. Nevertheless, in normal protein regulation, all likelihood curves should centre roughly around the same regulation value. In this sense, the normal case would be that all likelihood curves form a single cluster of such curves. The existence of two or more clusters is an indication of a deviating situation. The cause might be wrong measurements, the assignment of a peptide to a wrong protein or the special modifications of peptides. The main intention of clustering likelihood curves here is not find typical prototypes, but to group the curves and to discover cases where the curves are split into two or more clusters.

The application here involves small datasets that easily lead to unsuitable clustering results, since it can happen that single data objects – here likelihood curves – form a separated cluster. In order to reduce this effect, fuzzy clustering is applied.

Fuzzy clustering divides a dataset into a set of clusters and – in contrast to hard or deterministic clustering – a data object is not assigned to a unique cluster. In order to handle noisy and ambiguous data, membership degrees of the data to the clusters are computed. Most fuzzy clustering techniques are designed to optimise an object function with constraints. The most common approach is the so called probabilistic clustering with the objective function

$$f = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{m} d_{ij} \tag{6}$$

under the constraints

$$\sum_{i=1}^{c} u_{ij} = 1 \qquad \text{for all } j = 1, \ldots, n. \tag{7}$$

In this equation it is assumed that the number of clusters c is fixed. How to determine the number of clusters will be discussed later on. u_{ij} is the membership degree of data object x_j to the ith cluster. d_{ij} is some distance measure specifying the distance between data object x_j and cluster i, for instance the (squared) Euclidean distance of x_j to the ith cluster centre when the data objects are simple points, not likelihood curves as in the case of the investigations here. The parameter $m > 1$, called the fuzzifier, controls by how much clusters may overlap. The constraints (7) lead to the name probabilistic clustering, since in this case the membership degree u_{ij} can also be interpreted as the probability that x_j belongs to cluster i. The parameters to be optimised are the membership degrees u_{ij} and the cluster parameters that are not given explicitly here. They are hidden in the distances d_{ij}. Since this is a non-linear optimisation problem, the most common approach to minimise the objective function (6) is to alternatingly optimise either the membership degrees or the cluster parameters while considering the other parameter set as fixed. Assuming the cluster parameters and therefore the values d_{ij} as fixed, the best choice for the membership degrees is given by

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}} \right)^{\frac{1}{m-1}}}. \tag{8}$$

If $d_{ij} = 0$ for one or more clusters, one must deviate from (8) and assign x_j with membership degree 1 to one of the clusters with $d_{ij} = 0$ and choose $u_{ij} = 0$ for the other clusters i.

The update equation for the cluster parameters or cluster prototypes strongly depends on the type of cluster. For the specific case of likelihood curves, an algorithm is proposed in the following section.

Cluster validity measures are used to validate a clustering result in general and also to determine the number of clusters. In order to fulfil the latter task, the clustering might be carried out with different numbers of clusters and the one yielding the best value of the validity measure is assumed to have the correct number of clusters.

A straight forward validity measure is the objective function (6) itself. However, (6) will always decrease with increasing number of clusters. Therefore, if the number of clusters is determined based on (6), the procedure is as follows. The number of clusters c is increased step by step starting from $c = 1$ and (6) is evaluated each time. As long as increasing the number of clusters leads to a significant decrease of (6), the optimum number of clusters is still not reached. Once (6) starts to drop slowly when c is increased, c is too high.

There are other validity measures, such as the partition coefficient and the partition entropy [2]. Both these measures validate the clustering result based on the membership degrees only, without taking specific properties of the cluster prototypes into account.

The partition coefficient is defined by

$$\frac{\sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^2}{n}. \tag{9}$$

The higher the value of the partition coefficient the better the clustering result. The highest value 1 is obtained, when the fuzzy partition is actually crisp, i.e. $u_{ij} \in \{0,1\}$. The lowest value $1/c$ is reached, when all data are assigned to all clusters with the same membership degree $1/c$. This means that a fuzzy clustering result is considered to be better, when it is more crisp.

The partition entropy

$$-\frac{\sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}\ln(u_{ij})}{n} \tag{10}$$

is inspired by the Shannon entropy. The lower the value of the partition entropy, the better the clustering result. This means that, similar to the partition coefficient, crisper fuzzy partitions are considered to be better.

As mentioned before, there are many other validity measures for fuzzy clustering. However, they are not of interested here, since they assume the data to be points in $\mathbb{R}^p$ and not likelihood curves as in our case. For a general overview on fuzzy cluster analysis we refer to [3, 5].

4 Fuzzy Clustering of Likelihood Curves

Clustering of curves requires the definition of a distance measure, which is considerably more complex than comparing the positions of points in a coordinate system. Various kinds of distance measures for clustering curves are possible e.g. (i) maximum peak position, (ii) profile (width and height) of the curve or (iii) overlap of area.

Since likelihood curves give the distribution for the true (unknown) position of the maximum peak, it would not be advisable to take this value exclusively as the distance measure. Choosing the non-overlapping area of curves as distance measure combines the position of maximum peak on the one hand and the profile of the curve on the other hand. Furthermore, possible uncertainty concerning the exact position of the maximum peak is taken into account.

Thus, we have the objective function (6) and d_{ij} is given by

$$d_{ij} = 1 - \int_{-\infty}^{+\infty} min\{x_j(t), v_i(t)\}dt \tag{11}$$

with x_j = Peptide j and v_i = Prototype i.

Discretisation leads to

$$d_{ij} = 1 - \sum_{t=1}^{l_x} min\{x_j^{(t)}, v_i^{(t)}\}. \tag{12}$$

Areas under curves are normalised to 1, in consequence $0 \leq d_{ij} \leq 1$ and especially

$$d_{ij} = \begin{cases} 0 & \text{if complete overlap of Peptide } i \text{ and Prototype } j \\ 1 & \text{if no overlap of Peptide } i \text{ and Prototype } j. \end{cases}$$

Minimisation of the objective function f is done by generation of new prototype curves from the former prototype curves and the total amount of peptide curves (for details see Section 4.1). This process terminates when updating the prototypes yields no further decrease of f.

4.1 Generation of Prototypes

For the identification of the best partition of all peptide curves into c clusters (c fixed) c prototypes are initialised firstly. Subsequently, for all prototypes $i \in [1 \ldots c]$ and all peptides $j \in [1 \ldots n]$ the membership degrees u_{ij} are calculated by Eq. (8). The initialisation and update scheme for the cluster prototypes is described in detail in the following.

The listing below gives a general idea of the algorithm.

```
result[];
for ( $1 \leq c \leq n$ ){
    $f^{(p_{old})} = \infty$;
    p := initialise prototypes (curves, c);
    d := calculate distances (curves, p);
    u := calculate u (d);
    $f^{(p)}$ := evaluate cluster (u, d);
    while ( $|f^{(p)} - f^{(p_{old})}| > \varepsilon$ ){
        $p_{old}$ := p;
        $f^{(p_{old})} := f^{(p)}$;
        p := update prototypes (p, curves, u);
        d := calculate distances (curves, p);
        u := calculate u (d);
        $f^{(p)}$ := evaluate cluster (u, d);
    }
    result[c-1] = $p_{old}$;
}
return result;
```

Initialisation

In order to avoid unsuitable results based on an unfavourable initialisation step, we prefer repeated ($k = 3$) initialisation. Initialisation of c prototypes is done by randomly choosing c likelihood curves from the dataset, which are slightly modified by multiplying the centre with a fixed factor, and cutting off the edges when area $A = 1$ is reached. To obtain results as different as possible, it is important to make sure that different combinations of peptide curves are selected in all of the n initialisation steps.

Updating

The aim of repeated updating is the generation of a new set of prototypes from the previous set of prototypes. In the case of crisp clustering, where a likelihood curve l_j either belongs to a cluster i ($u_{ij} = 1$) or not ($u_{ij} = 0$), the update procedure is explained very easily. The more curves are overlapping at a position x, then the more the objective function for the clustering will be reduced when the prototype curve has a high value there as well. At first, areas with a high number of overlapping curves are added to the prototype. Step by step, less overlapping areas are added as well, and the procedure is finished when the area under the prototype likelihood curve reaches the value 1. Therefore, the new prototype likelihood curve is composed of those areas, where the most likelihood curves are overlapping. Besides the number of overlapping curves the weight w_{ij}

$$w_{ij}^{(t)} = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{m\ (t)} \tag{13}$$

strongly affects the development of a prototype i from likelihood curves l_j and the former prototype.

The update procedure for the actual prototype i is as follows: We assume that the horizontal axis is divided into T intervals of equal length l. Initially, all points $p_j^{(t)}$, that are part of the likelihood curve j, $0 \leq j \leq n$, are evaluated by application of equation (13) related to the considered prototype i. Therefore, points belonging to a likelihood curve which is similar to prototype i (high membership degree u_{ij}), are evaluated better than those which belong to a curve that is less overlapping with prototype i. Furthermore, the weight is increasing with every additional curve, which is overlapping with curve j in the considered interval.

A simple heuristic strategy to add the most interesting points to the new prototype is the following: All of the points are sorted in decreasing order with respect to their weights, regardless of their belonging to a special peptide curve. One after another, the points with the highest weights are added to the prototype likelihood curve under the constraint that every newly added point must be directly adjacent to the present dataset. This means that the x-coordinate of the new point x_p must not exceed the borders of the present interval of support of the partially constructed likelihood curve $[x_{min}, ..., x_{max}]$ for more than one interval length l ($x_{min} - l \leq x_p \leq x_{max} + l$).

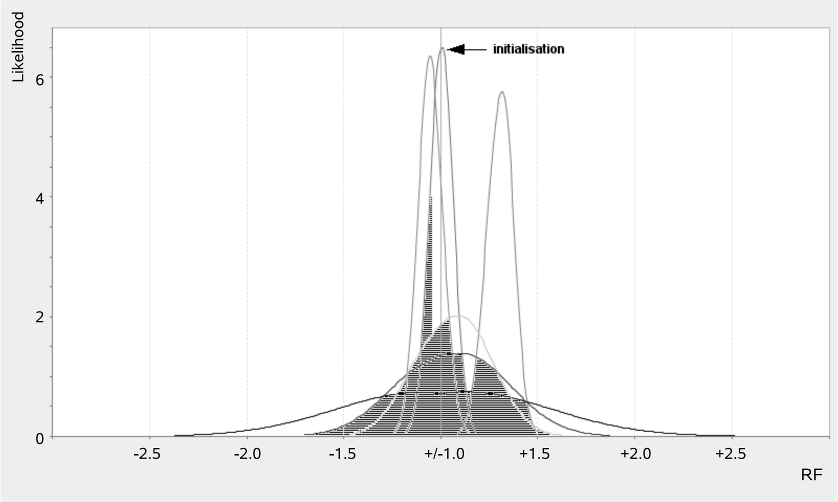


Fig. 7 Protein PCTK1_HUMAN: Resulting prototype after initialisation with the labelled curve and all possible updates. The prototype likelihood curve consists mainly of those areas where most of the data likelihood curves overlap

Figure 7 presents the resulting prototype after initialisation with the labelled curve and all possible updates. In this example we assumed that all six peptides build a single cluster ($c = 1$) and the prototype representing this cluster was calculated.

4.2 *Validity Measures*

As mentioned in Section 3, one way to determine the number of clusters would be to consider a prototype independent validity measure like the partition coefficient (Eq. (9)) or the partition entropy (Eq. (10)). Both measures are suitable for clustering larger datasets than those we are dealing with in this study. Typically, the number of likelihood curves to be clustered is between 2 and 10, and the majority of proteins is represented by less than 7 different curves. Due to these circumstances, neither of the well-established validity measures are suitable for our approach, due to the fact that a clear minimum and maximum respectively, can not always be found (for details see Example 2 in Section 4.3). Therefore, we prefer to use an alternative method as presented below to determine the optimal number of clusters.

The objective function f (Eq. 6) is minimised during calculation of the best clustering result for each c, $1 \leq c \leq n$ ($n =$ number of curves), and the result in each case is stored. As the result is decreasing with increasing c and always reaches its minimum for $c = n$, we can not regard the minimum value as the best one. We rather consider the differences (slope) Δ of $c - 1$ and c

$$\Delta = |f_{c-1} - f_c| \tag{14}$$

for every $c \in [2, \ldots, n]$.

In order to decide whether all likelihood curves can be represented by a single cluster, which is the normal and therefore the most frequent case, we compare the total area overlap of the resulting prototype and all n likelihood curves. If the overlap of every curve and the prototype is 50% of the curve's total area at least, we can assume that the similarity of all curves is significantly high and that they are represented by a single cluster. The threshold value of 50% was determined by analysis of curves, which form a single cluster by visual inspection.

4.3 Examples

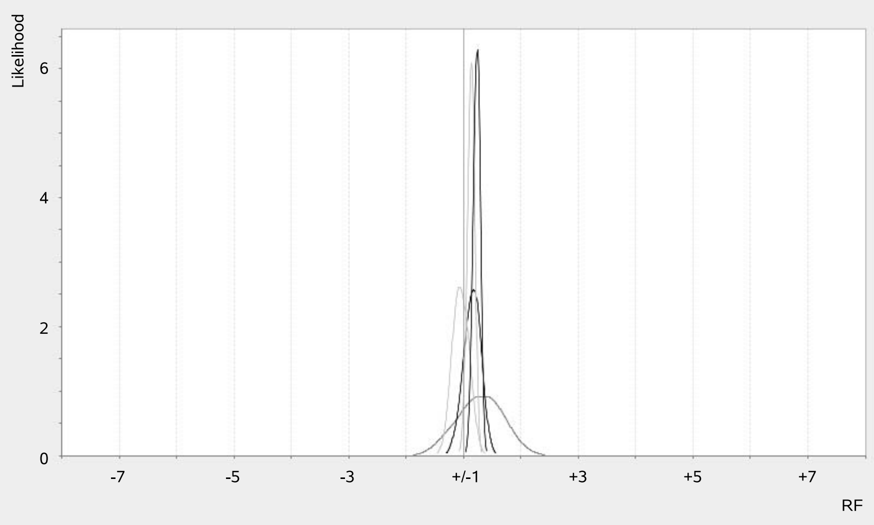
In order to demonstrate the difficulties described in Section 4.2 in determination of the optimal number of clusters c_{opt} and to present some results, we now give two examples. Every one of the following figures consists of a protein likelihood plot (top) and a combined illustration of the validity measures partition entropy, partition coefficient as well as the objective function.

First of all, we show a plot containing 5 curves (Figure 8). By visual inspection they are arranged in a single cluster. The calculated overlaps of all 5 curves with the final prototype for $c = 1$ are presented in Table 2. Since the overlap of all 5 curves with the prototype is greater than the half of each curves' total area, the algorithm terminates with the outcome that all peptide curves are to be considered as one single cluster. Validity measures, on the other hand, give no clear result. The outcome of the investigation of partition entropy is quite clear and proposes 2 clusters, partition coefficient analysis slightly tends towards 1 cluster. The objective function's slope finally gives no information about the quality of a single cluster in principal and is not to be regarded in this case.

Table 2 Overlapping areas of likelihood curves and resulting prototype in the case of $c = 1$. As the overlaps are at least half of the curve's total area in every case, the clustering algorithm returns that one single cluster was found

Curve 1	Curve 2	Curve 3	Curve 4	Curve 5
59%	65%	86%	54%	53%

The likelihood plot of the second example (Figure 9) clearly shows 2 clusters, each containing 2 curves. The results of the analysis of area overlaps of peptide curves and prototype in the case of $c = 1$ are given in Table 3. As 2 of the 4 curve overlaps are less than 50%, the existence of one single cluster can be excluded. Partition entropy as well as the slope of objective function yield the result that the data builds 2 clusters. However, the interpretation of the partition coefficient is not easy,



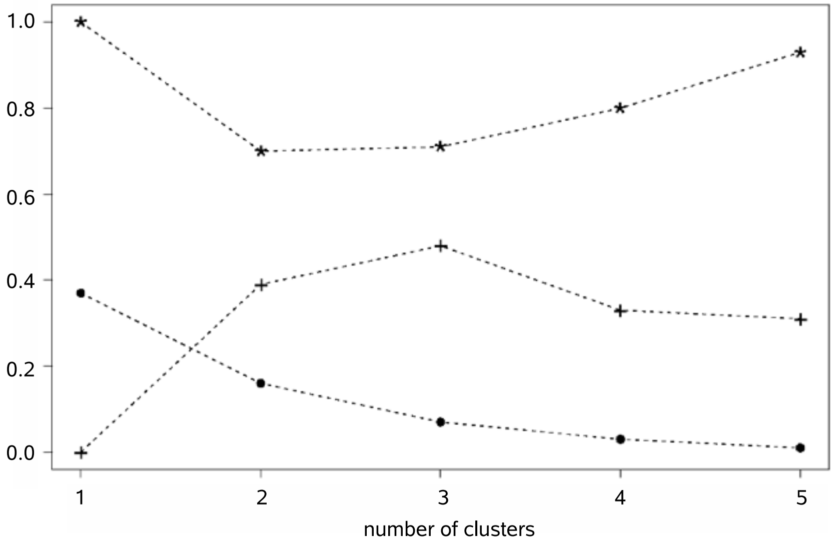
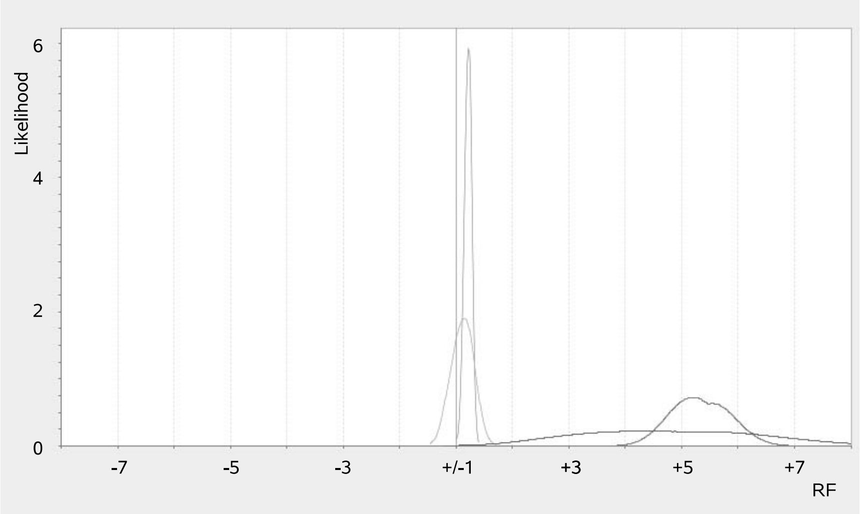


Fig. 8 Protein LYN_HUMAN: Top: 5 peptide curves clustering into 1 cluster. Bottom: partition entropy giving $c_{opt} = 2$ (+), objective function resulting $c_{opt} = 2$ (•), partition coefficient giving $c_{opt} = 1$ ($*$)



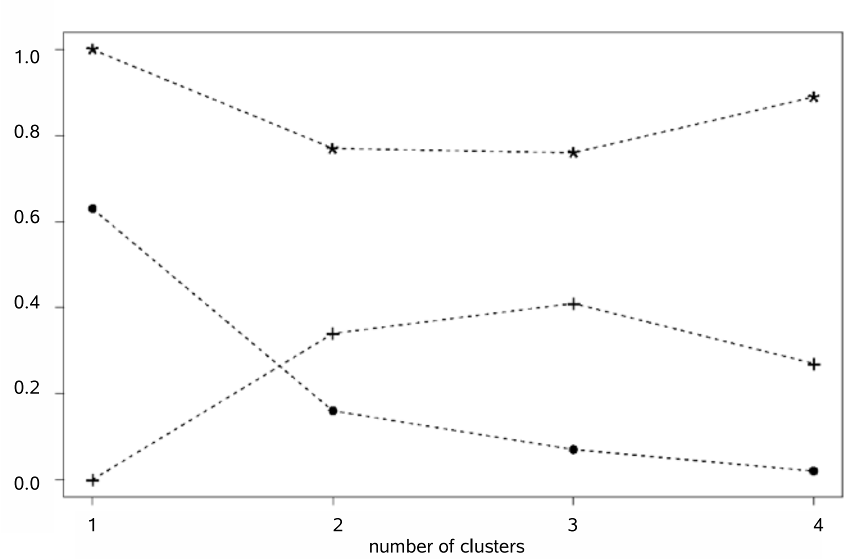


Fig. 9 Protein MK01_HUMAN: Top: 4 peptide curves clustering into 2 clusters. Bottom: partition entropy giving $c_{opt} = 2$ (+), objective function resulting $c_{opt} = 2$ (•), partition coefficient ranging between $c_{opt} = 1$ and $c_{opt} = 2$ ($*$)

Table 3 Overlapping areas of likelihood curves and resulting prototype in the case of $c = 1$. As the overlaps are not at least the half of the curve's total area in every case, the clustering algorithm returns that more than one single cluster was found

Curve 1	Curve 2	Curve 3	Curve 4
90%	58%	2%	0%

as its value is between 1 and 2 clusters. What is the reason for this ambiguous behaviour? If the optimal number of clusters were $c_{opt} = 2$ as in this case, partition coefficient $_{c=2}$ must be higher than partition coefficient $_{c=3}$. Since this is an example with $n = 4$ likelihood curves in a total of 3 clusters represented by 3 different prototypes means that nearly every curve has its own prototype and the partition coefficient $_{c=3}$ will rarely be significantly less than partition coefficient$_{c=2}$. In any case, samples resulting with partition coefficient $_{c=3} \geq$ partition coefficient$_{c=2}$ do occur. This kind of problem is caused by the very low number of clustering objects. Depending on initialisation and the further design of prototypes both partition coefficient and partition entropy can be affected. As slope of objective function on the other hand is a more reliable measure we prefer to use this one instead of the established validity measures in our special case.

5 Conclusions

We have introduced a noise model and derived likelihood curves for the visualisation of regulatory information and the analysis of the robustness of quantitative LC-MS/MS data after iTRAQ$^{\text{TM}}$-labelling. Furthermore, we presented an approach for fuzzy clustering of likelihood curves, in order to reveal erroneous measurements, the assignment of a peptide to a wrong protein or special modifications of peptides. The aim of clustering likelihood curves is to group the curves and to discover proteins where the peptide curves are split into two or more clusters. Upcoming problems concerning the determination of the number of clusters by means of well-established validity, means partition coefficient and partition entropy were solved by definition of new criteria specific to the available data for the detection of both a single cluster and multiple clusters as well as the number of clusters in the last case.

By the means of the workflow – estimation of the noise inherent in quantitative LC-MS/MS data analysed by using the iTRAQ$^{\text{TM}}$ method, calculation and subsequently clustering of likelihood curves – advanced systematic analyses of those data is possible.

The presented approach is easily transferable to other noisy data, if the noise can be specified by a noise model. Then, after once-only estimation by means of a suitable biological sample, all derived applications, like calculation and clustering of likelihood curves for the visualisation of regulatory information and robustness, can be used.

References

1. Bäck, T.: Evolutionary Algorithms in Theory and Practise. Oxford University Press, Oxford (1996)
2. Bezdek, J.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
3. Bezdek, J., Keller, J., Krishnapuram, R., Pal, N.: Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. Kluwer, Boston (1999)
4. Boehm, A.M., Pütz, S., Altenhöfer, D., Sickmann, A., Falk, M.: Precise protein quantification based on peptide quantification using itraq. BMC Bioinformatics 8, 214 (2007)
5. Höppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis. Wiley, Chichester (1999)
6. Hu, J., Qian, J., Borisov, O., Pan, S., Li, Y., Liu, T., Deng, L., Wannemacher, K., Kurnellas, M., Patterson, C., Elkabes, S., Li, H.: Optimized proteomic analysis of a mouse model of cerebellar dysfunction using amine-specific isobaric tags. Proteomics 6(15), 4321–4334 (2006)
7. Hundertmark, C., Fischer, R., Reinl, T., May, S., Klawonn, F., Jnsch, L.: Ms-specific noise model reveals the potential of itraq^{TM} quantitative proteomics (2008) (submitted)
8. Klawonn, F., Hundertmark, C., Jansch, L.: A maximum likelihood approach to noise estimation for intensity measurements in biology. In: Proc. Sixth IEEE International Conference on Data Mining Workshops ICDM Workshops 2006, pp. 180–184 (2006)
9. Lin, W.-T., Hung, W.-N., Yian, Y.-H., Wu, K.-P., Han, C.-L., Chen, Y.-R., Chen, Y.-J., Sung, T.-Y., Hsu, W.-L.: Multi-q: a fully automated tool for multiplexed protein quantitation. J. Proteome. Res. 5(9), 2328–2338 (2006)
10. Ong, S.-E., Blagoev, B., Kratchmarova, I., Kristensen, D.B., Steen, H., Pandey, A., Mann, M.: Stable isotope labeling by amino acids in cell culture, silac, as a simple and accurate approach to expression proteomics. Mol. Cell. Proteomics. 1(5), 376–386 (2002)
11. Pierce, A., Unwin, R.D., Evans, C.A., Griffiths, S., Carney, L., Zhang, L., Jaworska, E., Lee, C.-F., Blinco, D., Okoniewski, M.J., Miller, C.J., Bitton, D.A., Spooncer, E., Whetton, A.D.: Eight-channel itraq enables comparison of the activity of 6 leukaemogenic tyrosine kinases. Mol. Cell. Proteomics (2007)
12. Ross, P.L., Huang, Y.N., Marchese, J.N., Williamson, B., Parker, K., Hattan, S., Khainovski, N., Pillai, S., Dey, S., Daniels, S., Purkayastha, S., Juhasz, P., Martin, S., Bartlet-Jones, M., He, F., Jacobson, A., Pappin, D.J.: Multiplexed protein quantitation in saccharomyces cerevisiae using amine-reactive isobaric tagging reagents. Mol. Cell. Proteomics. 3(12), 1154–1169 (2004)
13. Walsh, C.T., Garneau-Tsodikova, S., Gatto, G.J.: Protein posttranslational modifications: the chemistry of proteome diversifications. Angew. Chem. Int. Ed. Engl. 44(45), 7342–7372 (2005)

A Hybrid Rule-Induction/Likelihood-Ratio Based Approach for Predicting Protein-Protein Interactions

Mudassar Iqbal, Alex A. Freitas, and Colin G. Johnson

Abstract. We propose a new hybrid data mining method for predicting protein-protein interactions combining Likelihood-Ratio with rule induction algorithms. In essence, the new method consists of using a rule induction algorithm to discover rules representing partitions of the data, and then the discovered rules are interpreted as “bins” which are used to compute likelihood ratios. This new method is applied to the prediction of protein-protein interactions in the *Saccharomyces Cerevisiae* genome, using predictive genomic features in an integrated scheme. The results show that the new hybrid method outperforms a pure likelihood ratio based approach.

1 Introduction

Protein-protein interactions are involved in almost every cellular function, from DNA replication and protein synthesis to regulation of metabolic pathways [1]. Proteins interact with each other by physically binding themselves or with other molecules in the cell and form larger complexes to perform specific cellular functions. Hence, the study of protein-protein interactions is of utmost importance to understand their functions [7, 2], and detailed information about the interactions of proteins can have potentially very useful applications, e.g., predicting disease-related genes by looking at their interactions [28] as well as a potential use in developing new drugs that can specifically interrupt or modulate protein interactions [41]. Also, the study of these interactions at the genomic level can help understanding the large scale organization and features of the underlying network and the role of individual proteins within the network [46].

Consequently a number of experimental techniques for determining protein-protein interactions have been developed [39, 20, 12, 17]. Unfortunately the

Mudassar Iqbal, Alex A. Freitas, and Colin G. Johnson
Centre for Biomedical Informatics and Computing Laboratory, University of Kent, Canterbury, U.K.
e-mail: mi26,a.a.freitas,c.g.johnson@kent.ac.uk

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 623–637.
springerlink.com

experimental determination of the interaction network of even very simple organisms is difficult and potentially erroneous, and the overlap among the interactions determined by different such techniques is very low [46, 42]. Hence, there is a clear motivation to develop new computational methods which can use data integrated from several genomic sources, as it is done in this work, as explained below. Many experimental and computational methods for the prediction of protein-protein interactions are discussed in recent reviews [36, 37, 41].

1.1 Computational Prediction of Protein-Protein Interactions

The purpose of computational methods is to predict unknown protein interactions using the relevant genomic information available, i.e., computational methods typically try to predict protein interaction by using data produced by other genomic techniques such as gene expression, localization etc, which are indirectly related to protein interactions. A variety of computational methods have been investigated for this problem so far. Many methods infer interactions from a single type of genomic data. For example, [3] and [4] address the question whether protein interactions can be predicted directly from the primary structure and associated data. Given a database of interacting proteins, they develop a machine learning system (Support Vector Machine) trained to recognise the potential interactions based solely on the primary structure and the associated physicochemical properties.

Another well-known method is called the Rosetta Stone Method. In this method, Marcotte et al. [23] find and exploit a very interesting observation that:"some pairs of interacting proteins have homologs in another organism fused into a single protein chain (Rosetta stone)". Other biological hypotheses used for prediction of protein-protein interactions include similarity in phylogenetic profiles [11] and co-evolution of interacting partners [15, 16].

Another approach consists of casting the protein-protein interaction prediction problem as a type of combinatorial optimization problem (Satisfiability) by looking at the domain (conserved evolutionary units within the proteins) assignments of interacting and non-interacting protein pairs and then using a combinatorial optimization method to solve it. In [18] a particle swarm optimization method, a relatively new type of computational intelligence algorithm, was used to infer domain-domain interactions and then use the inferred domain-domain interactions to predict new protein-protein interactions. Yet another approach consists of analyzing protein-protein interaction data to infer domain-domain interactions using graph-theoretical belief propagation methods [19].

Also, there is a whole group of methods in which information from different genomic features is combined to predict interactions. Such methods are here called "Integrative Methods" . For instance, in [45], the authors build an integrative model using a kernel based method combining many heterogenous data sets and present a supervised learning approach for prediction of protein interactions. Jansen et al. in [21] formulate a Bayesian framework for combining different types of data and predict genome wide interactions in *Yeast*. The basic idea is that given certain features corresponding to protein pairs under consideration and their class attribute (interacting or non-interacting), one can estimate the likelihood of interaction for a given feature, and overall likelihood is estimated using a naive Bayesian formulation by assuming independance among all the features. Rhodes et al.[33] extend this Bayesian approach for predicting protein interaction to the human genome. The general idea of all the integrative methods is that one could combine various relatively weak features in a setting in which overall prediction is boosted by this integration of data. Some interesting observations are drawn in [22] regarding this data integration for protein interaction prediction. A detailed analysis of this data integration using different classifiers is researched in [5].

1.2 Overview of the Proposed Method

Our work is partly inspired by the work done by [21], in which they proposed a Bayesian method using the MIPS (Munich Information center for Protein Sequences [24]) complexes catalog as a gold standard for positive interactions, and a list of proteins in separate sub-cellular compartments as negative interactions, as there is no particular data set of experimentally determined non-interactions. They integrate multiple genomic data corresponding to protein pairs, including correlation in expression levels, functional similarity based measure, etc., as well as other experimental data about protein interactions, as predictive features for these positives and negatives. We use many of the protein pair features used in [21] and a subset of their gold standard non-interactions to conduct a data mining experiment in order to analyze the effect of hybridizing simple naive Bayesian style likelihood based method with some rule induction algorithms. Rule induction algorithms learn classification rules given the predictive features as well as the class attribute of a set of examples (protein pairs in this case). Those learned rules can be used to predict unknown protein interactions. We first analyze a simplified version of the naive Bayes classification method without using any prior information and analyze its behavior for different possible values of sensitivity and specificity of prediction. Then we combine that simplified naive Bayes formulation with another data mining algorithm, namely a rule induction algorithm which learns *IF-THEN* type classification rules from data.

In essence, we propose a new hybrid approach where we use the partitioning of the data corresponding to the induced rules as “bins” from which likelihood ratios are computed and used to classify the data. We present a ROC (receiver operating characteristic) curve analysis of results obtained using different threshold levels on the calculated likelihood values. Since these rules consist of multiple antecedents coping with attribute interactions, the bins defined by these rules should give us a better insight as compared to the uniform binning of attributes used in general naive Bayesian methods. We have applied this hybrid method to a specific biological application here, e.g., prediction of protein-protein interactions in the yeast *S. Cerevisiae* using multiple genomic features, but the underlying principles of the method are not application domain dependent, and indeed it can be applied to a wide range of classification problems in different application domains.

1.3 Organisation

The chapter is organized as follows. Section 2 discusses the background on rule induction algorithms for classification. Section 3 details the protein interaction data and data related to different attributes used. Section 4 explains our method, starting with a brief introduction to Naive Bayes for classification and a rule discovery algorithm, the *PART*[1] method for classification; and then proposes a new hybrid method combining features of both techniques. Comparative results based on the ROC analysis are presented in Section 5. Finally, Section 6 concludes the chapter.

2 Classification Rule Discovery Algorithms

Classification is one of the major data mining tasks. Given the data (examples) with the predictive attributes and class labels (e.g. interacting or non-interacting in a protein pair's case), the task of classification amounts to find relationship(s) among the attributes and class labels. These relationships can be in the form of, for instance, *IF-THEN-ELSE* type rules, decision tree or conditional probabilities depending upon which approach is used for building the model [43]. A classification model is built using the training data, i.e., with class value known, and that learned model's quality is then tested on the test data, i.e., where the class value is absent.

Classification rules are one of the popular data mining approaches mainly because of their comprehensibility, by representing the gained knowledge in a form which is intuitive to human understanding. These rules have two parts, i.e. the rule antecedent – which is a conjunction of multiple conditions on the predictor attributes – and a rule consequent-which is the prediction of class attribute based upon the conditions in the antecedent. Conditions over individual attribute values potentially involve all relational operators.

$$IF cond_1 AND cond_2 ... THEN class$$

There are many approaches to building models involving rule sets for a classification problem. One most common and widely used approach is the separate-and-conquer approach, which we will discuss in some detail in the next section. Another popular classification method is the divide-and-conquer technique by building decision trees [31]. In the work below we describe a rule induction algorithm that uses aspects of both of these approaches. Therefore, we begin by reviewing the basic concepts of these two methods.

[1] PART builds rule sets using partial decision trees.

2.1 Separate-and-Conquer Approach

One of the two main approaches to rule induction is the *separate-and-conquer* approach. This approach was originally devised in [25] with the name *covering* strategy, whilst the term separate-and-conquer was introduced by Pagllao&Haussler [29]. Many different variants on this approach, designed to tackle different problems and data types, have been implemented; the review by Fürnkranz [10] gives an overview.

The general idea of the separate-and-conquer approach begins with an induction of a rule, via some rule induction algorithm, on the entire dataset. The examples that are correctly classified by this rule are then removed from the dataset, and the rule induction algorithm applied to this reduced dataset. An example is said to be correctly classified by a rule when the example's attribute values satisfy the conditions in the rule's antecedent and the example's class is the same as the class predicted by the rule's consequent. This process of rule induction and removal of covered training examples is repeated until the dataset is empty. In this way each example in the training data will be covered by at least one rule.

This approach has been used with a number of rule representations, i.e., the allowed structure of the antecedent in the *IF-THEN* rule. In early work on this approach [26] the antecedent of the rules is a simple relation between attribute and value; for example, a threshold for a numerical value. In other approaches, more sophisticated representations are allowed, for example in FOIL [32] PROLOG relations are used. A more sophisticated approach is to allow the representation to expand when needed [40]; a number of approaches to this are reviewed by Fürnkranz [10].

Many different approaches have been used for the rule induction mechanism itself. These include both deterministic methods such as hill climbing [32] and beam search [27], and stochastic methods such as evolutionary algorithms [14].

One danger with these methods is that they can suffer from *overfitting*, where the model is fitted too specifically to the (noisy) training data set, and is therefore unable to generalise well to the test data set (unseen during training). Methods for tackling this problem revolve around the idea of *pruning* the rule set, either by removing whole rules, or by simplifying the precedent of the rule [35]. Such pruning methods fall into two main types: pruning methods that operate whilst the learning process is running (so called *pre-pruning* methods), and *post-pruning* methods that process the rule set after it has been generated. These methods are reviewed by Fürnkranz [10].

2.2 Divide-and-Conquer Approach

In the previous section, the dataset was split by *instances*, each application of the rule induction algorithm removing some instances from the dataset. By contrast, the divide-and-conquer approach splits the attribute space as it works. The canonical representation used in the divide-and-conquer approach is the *decision tree*.

An example of a decision tree is given in Figure 1. To predict the class of an unseen data instance using the tree, the tree is worked from the root. The algorithm evaluates the condition at the root node, and then moves on the the left or right child node depending on whether the result of evaluating that condition is true or false. This process is repeated until a leaf node is found; the leaf node names the class that should be assigned to that instance.

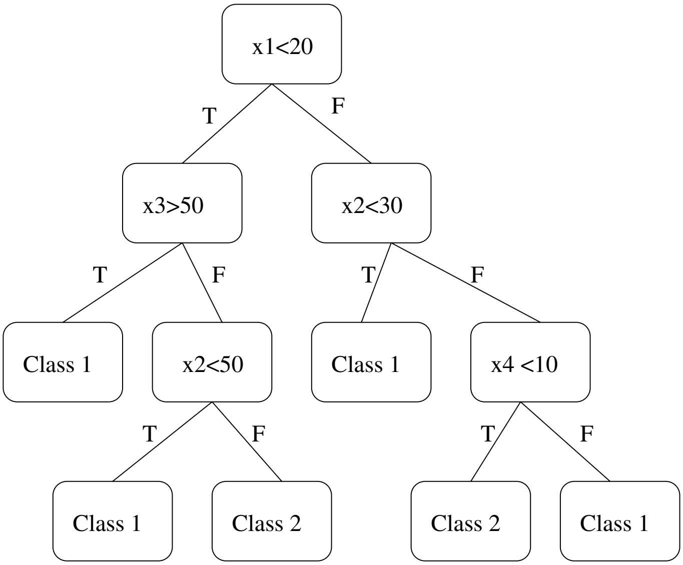


Fig. 1 An example of a decision tree

A number of methods have been devised for the induction of decision trees from data sets. The most widely used methods are those based on *information gain*, first introduced by Quinlan [30, 31]. This begins by constructing putative tree “stumps” [43], based on a number of options for the condition in the root node (how these options are constructed is algorithm and data-type specific). The training set is then distributed between the edges adjacent to this node based on this criterion, and a measure of the balance of classes associated with each of these edges is calculated. This measure is highest when an edge contains only one class (as there is no more decision to be made) and lowest when there is an equal balance of classes (as no information has been provided by the consideration of that condition). Based on this measure, the condition that maximises this information gain is chosen. This is then recursively repeated for lower levels of the tree, until one of the following conditions is satisfied: all classes are classified correctly (i.e. there are no “impure” edges); no more non-contradictory conditions can be created; or some algorithm-specific criterion for the simplicity of representation is satisfied (to avoid overfitting).

3 Protein Interaction Data and Predictive Features

We use four different features which were described as highly predictive features in the detailed analysis done by Lu et al. [22]. These features are explained below. All these features have been downloaded from supplementary material available with [22] and available online at `http://networks.gersteinlab.org/intint`.

- mRNA Co-expression (COE): Based on the hypothesis that interacting proteins have correlated expression profiles [21, 22, 13], this feature seems promising for the prediction of protein-protein interaction.
- MIPS Functional Similarity (MIPS): Interacting proteins often function in the same biological process [22]. The data associated with this feature was extracted from the MIPS functional catalog.
- GO[2] Functional similarity (GOF): This data is based on a similar hypothesis as the MIPS data, but is created using the GO functional classification scheme. The details of preparation of the data are given in [21] and [22].
- Marginal Essentiality (MES): This is a quantitative measure of the importance of non-essential genes to a cell [47] and it is based on the Marginal-Benefit Hypothesis that many non-essential genes make a significant but small contribution to the fitness of the cell [38].

Of course there are many other genomic features available like essentiality, data derived from Rosetta stone method, etc; but most such features are very scanty, i.e., very few protein pairs have known values for these features, or their predictive power is very low as compared to the above mentioned highly predictive features. Hence, in this work we only use the above mentioned highly predictive features along with some high confidence gold standard interacting and non-interacting pairs of proteins.

We obtained the *S. Cerevisiae* protein interaction data from DIP (Data base of Interacting Proteins [34, 44]). We obtained nearly 5000 high confidence positive interactions in DIP, called CORE, which is a subset of the total number of reported protein interactions in DIP. Negative interactions are hard to find. As used by many researchers in this field we consider a protein pair as a negative example (i.e., the proteins in question do not not interact) if the proteins in the pair are not in the same cellular compartment [21, 22]. This gives us many hundred thousands of protein pairs which are not co-localized. As there are too many negative examples found in this way, we keep only a small subset of those. We obtained gold standard negatives from [22]. For both positive and negative gold standard data, we keep only those pairs with complete information, i.e., with no missing values in the four predictive features which we use. After this preprocessing we end up with 2122 positive interactions (positive examples) and 5656 negative ones in our gold standard set.

[2] Gene Ontology.

4 A New Hybrid Rule Induction/Likelihood-Ratio Based Method

In this section we will first discuss two different approaches which can be used for prediction of protein-protein interactions. First we will describe a naive Bayesian formulation which is based on the estimation of likelihood values of interactions given the predictive features, followed by a discussion of rule induction algorithms which output a classification rule set. Then we will describe a hybrid approach which integrates both rule induction algorithms and likelihood ratios drawn from the naive Bayesian approach.

4.1 From Naive Bayes to a Likelihood Based Approach for the Prediction of Protein-Protein Interactions

The Bayesian approach is widely used in inference problems in many different areas, including several types of bioinformatics problems. Jansen et al. [21] and Rhodes et al. [33] used a form of Naive Bayes classifier to predict protein-protein interactions by combining multiple features. Given a data set of interacting proteins considered as positives, and a set of protein pairs separated in different cellular compartments considered as negatives, prior odds are defined as:

$$O_{prior} = \frac{P(pos)}{P(neg)} = \frac{P(pos)}{1-P(pos)} \tag{1}$$

Where $P(pos)$ and $P(neg)$ is the fraction of positives and negatives respectively among all pairs of proteins in the training data. The posterior odds that a pair of proteins interacts given the predictive features $f_1 ... f_n$ is:

$$O_{posterior} = \frac{P(pos|f_1 ... f_n)}{P(neg|f_1 ... f_n)} = O_{prior} * L(f_1 ... f_n) \tag{2}$$

$L(f_1 ... f_n)$ is the likelihood ratio and is defined as:

$$L(f_1 ... f_n) = \frac{P(f_1 ... f_n|pos)}{P(f_1 ... f_n|neg)} \tag{3}$$

Making the Naive Bayes assumption that the predictive features are independent from each other given the class (positive or negative), the likelihood ratio can be easily calculated as the product of individual likelihood ratios for each feature f_i as per Eq.4.

$$L(f_1 ... f_n) = \prod_{i=1..n} L(f_i) = \prod_{i=1..n} \frac{P(f_i|pos)}{P(f_i|neg)} \tag{4}$$

$L(f_i)$ is calculated as the fraction of positives having feature f_i divided by the fraction of negatives having feature f_i. As we are using a relatively small subset of total interactions and non-interactions and a reliable estimate of prior odds does not seem to be available, we do not use the prior odds at all in this formulation, and hence the posterior odds are the same as the likelihood ratio. Since the prior odds are not used, we analyze the predictive accuracy obtained for different threshold cutoffs of likelihood ratio values, instead. Hence, in this paper, we use a likelihood based approach for the prediction of protein interactions.

4.2 Generating Classification Rules for Protein-Protein Interaction Prediction

A popular type of data mining methods consist of building predictive models in the form of *IF-THEN* classification rules. More precisely, each rule has the form:

IF (condition(s) on attribute value(s)) THEN (class value)

Hence, each rule represents a relationship between the predictor attributes (features) and the goal attribute. Rules are discovered using the training set. The discovered rules are then used to predict the class value of examples in the test set, unseen during training [9]. Rule induction methods are known to present the knowledge discovered from the data in a comprehensible form to the users. Such comprehensible rules can be very helpful for the domain experts, for example biologists in our case, who can validate the discovered rules and potentially get new insight about the data. The discovered rules also have the potential to represent new knowledge about the problem at hand.

A variety of approaches exist for learning accurate and comprehensible rules from the data [43]. One line of research is to begin with building a decision tree and then transform it into a set of rules [31]. However, in the literature the term rule induction is often used to refer to an algorithm which discovers rules somewhat more flexible than a decision tree, in the sense that the discovered or induced rules cover data space regions that can have some overlap (unlike the leaf nodes of a decision tree, which represent non-overlapping data space regions). Most rule induction algorithms use the previously discussed separate-and-conquer approach, which tries to determine the most powerful rule that underlies the data by sequentially adding conditions on the attributes to the rule, separates out those examples that are covered by the rule and repeats the procedure on the remaining examples [6].

For the problem at hand, we use a method called PART [8] for the classification of protein-protein pairs (examples or data instances) into interacting or non-interacting. PART involves features of both decision tree building and rule induction algorithms – both of which were reviewed above. PART is available for use in the freely available data mining package WEKA[3] [43]. The basic idea of this method is that it uses the separate-and-conquer strategy, as in the case of rule induction algorithms, in that it builds a rule, removes the examples it covers and continues creating rules for the remaining examples until none are left. But it differs from most rule induction algorithms in the way a rule is induced. To build a single rule, first a pruned decision

[3] Waikato Environment for Knowledge Analysis.

tree is built for the current set of examples. Then the leaf with the largest coverage is made into a rule, and the tree is discarded. This process is iteratively repeated until all training examples are covered by the induced set of rules. The details about the PART method and its comparison to other competing methods are in [8].

4.3 Classification Rule Discovery as a Binning Method for a Likelihood-based Approach

When using the rule induction method described in previous subsection, most of the discovered rules contain conditions on multiple predictor attributes. For example, the following rule containing conditions on two attributes (GO and MIPS as defined in Section 2) and predicting class 0 (negative interaction).

$$IF\ (\ GO \geq 3.85\ AND\ MIPS \geq 5.45\ AND\ MIPS \leq 6.15\ )\ CLASS = 0$$

There will be some negative examples satisfying this rule as well as some (perhaps small in number) positive examples. Unlike the Naive Bayes method, each discovered rule represents an interaction among the attributes in the rule's antecedent (since all attributes in that antecedent have to be satisfied, in order for an example to satisfy the rule). We can view each of these rules as a multiple-attribute binning of the data. This allows us to compute the likelihood in a way conceptually similar to Eq.3, i.e., the fraction of positive examples satisfying this rule antecedent divided by the fraction of negative examples satisfying the antecedent of this rule, but with the difference that, instead of computing a likelihood for each individual feature, we compute a likelihood for each "bin", i.e., each conjunction of the attribute values in a rule antecedent. Of course the bins and corresponding likelihoods are computed using rules discovered from the training data. We then evaluate this hybrid predictor's performance on the test set using a ROC curve. In other words, after having the rules, or these multi-attribute bins, we calculate their likelihood ratios and predictive accuracy by putting different thresholds on the minimum value of the likelihood ratio required to assign an example to the positive class. In this way we can analyze the whole range of threshold values like in the case of the Naive Bayes method, instead of the hard classification done by a stand alone rule based method. We analyze the effects of these multi-attribute bins/rules against the assumption of the Naive Bayes method which assumes independence among the attributes given the class.

5 Results and Discussion

We present here a ROC (receiver operating characteristic) curve analysis of the results obtained using a likelihood-based approach as explained in section 4 and using our hybrid approach based on a rule learner combined with likelihood ratio test. A ROC curve graphically depicts the performance of a classifier at different levels of thresholds that we put on the minimum likelihood for prediction of positive interaction in this two class classification problem. In other words, for a given threshold value t, a test example (protein pair) is predicted to have interaction (positive class) if and only if the value of likelihood (Eq. 3) is greater than or equal to t. This kind of analysis gives us an opportunity to evaluate the classifier not just by the total number of classification errors it makes, but rather allows us to analyze what is the tradeoff among two different types of errors, i.e., false positive predictions and false negative predictions. It plots true positive rate (sensitivity) vs false positive rate (1-specificity), where each point in the curve belongs to a particular threshold on the likelihood value. In this way we can analyze the effect of different thresholds on predictive accuracy instead of analyzing the effect of a single threshold using prior odds.

We use the 10-fold cross validation procedure [43] in all experiments reported here. Both positive and negative interaction data along with the predictive features is divided into ten equal folds, respectively. For each experiment, we divide the data (for both positive and negative classes along with their features separately) randomly in ten equal folds. Each time we use nine out of ten folds as training and the remaining one fold as a test. This process is repeated ten times, each time using a different fold as the test set. Likelihood values estimated during the training run are used to predict protein-protein interactions in the test examples. Sensitivity and specificity are defined by Eq. 5 and 6,

$$Sensitivity = \frac{TP}{TP+FN} \tag{5}$$

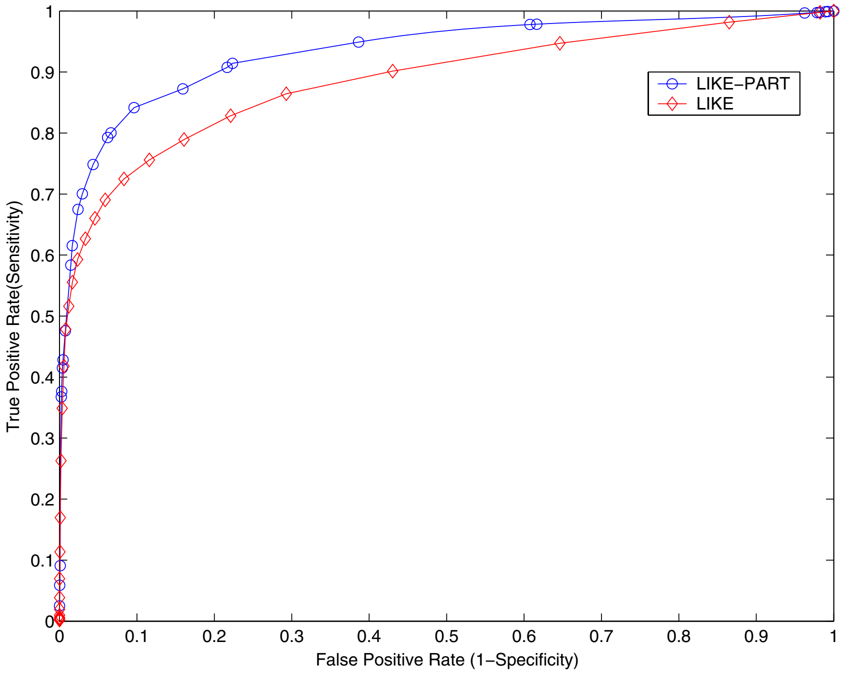


Fig. 2 ROC Curve for LIKE-PART and Pure Likelihood-based Approach (LIKE)

Table 1 Results for maximum value of accuracy for both methods

$Method$	$LogLR_{cut}$	TPR	$FPR(1-Spec)$	$Sen*Spec$	Acc	TP/FP
LIKE-PART	0.49	0.748	0.0433	0.716	0.8998	6.48
LIKE	0.6	0.66	0.0457	0.63	0.874	5.41

$$Specificity = \frac{TN}{TN+FP} \tag{6}$$

Where TP,TN,FP and FN are the number of true positives, true negatives, false positives and false negatives, respectively. A ROC curve for a good classifier will be as close as possible to the upper left corner of the graph, with a large area under the curve. Fig.2 shows the ROC curves for pure likelihood-based approach (hereafter called LIKE) and the hybrid method (hereafter called LIKE-PART, i.e., Likelihood based classifier using PART for finding rules/bins). The corresponding areas under the curve are 0.8862 and 0.9325, showing a better predictive performance of the LIKE-PART hybrid.

We can see from the Figure 2 that taking into account the multi-attribute binning or the rules produced by the base rule learner has enhanced the overall performance of the classifier significantly, even though the features in this data are not so well correlated, as reported in [22]. Table 1 reports the results for the likelihood cutoffs which correspond to maximum predictive accuracies for both methods. A statistical significance test performed on the accuracy values over ten folds for these likelihood cutoffs gives a p-value of 0.0000017, which indicates that LIKE-PART outperforms the LIKE method very significantly.

6 Conclusions

In this work, we have addressed a challenging and important bioinformatics problem, namely the prediction of protein-protein interactions using a hybrid data mining technique combining rule induction methods with likelihood ratio based classifiers. We used integration of different genomic features for a small data set and implemented two versions of a likelihood ratio based classifier. We did not use any prior odds, but rather used only likelihood ratio and presented a range of results using a ROC curve for different thresholds of the likelihood values used as a minimum value for the prediction of positive examples. We proposed a new hybrid method which used a known Rule Induction algorithm (*PART*) to induce rules from the training set taking into account possible attribute interactions and then interpret each rule as a bin for the likelihood based classifier. Since these bins were produced by taking into account attribute interaction, they avoid the unrealistic assumption of independence between attributes that is made by a pure likelihood based classifier. Then we compared the ROC curve of this new hybrid PART/Likelihood-based method with the ROC curve of the pure likelihood-based method and we observe that the hybrid method significantly improves as an overall classifier. Also, in the proposed method we can use different levels for likelihood value cutoff for final prediction, which gives us a more general setting where one can go for different levels of sensitivity and specificity.

We have evaluated this collaborative technique in the specific problem of predicting protein-protein interactions using genomic features, but the basic idea behind the technique, i.e., using induced rules as multi-attribute bins for the likelihood ratio based classifier, can be used for other classification problems easily, since it is independent of the application domain.

References

1. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P.: Molecular Biology of the Cell, 2nd edn. Garland, New York (1989)
2. Aloy, P., Russell, R.B.: Structural systems biology: modelling protein interactions. Nat. Rev. Mol. Cell. Biol. 7(3), 188–197 (2006)
3. Bock, J.R., Gough, D.A.: Predicting protein-protein interactions from primary structure. Bioinformatics 17(5), 455–460 (2001)
4. Bock, J.R., Gough, D.A.: Whole proteome interaction mining. Bioinformatics 19(1), 125–135 (2003)
5. Browne, F., Asuaje, F., Wang, H., Zheng, H.: An assessment of machine and statistical learning approaches to inferring networks of protein-protein interactions. Journal of Integrative Bioinformatics 3(2) (2006)
6. Cohen, W.W.: Fast effective rule induction. In: Proceedings of the 12th International Conference on Machine Learning, pp. 115–123. Morgan Kaufmann, San Francisco (1995)
7. Eisenberg, D., Marcotte, E.M., Xenarios, I., Yeates, T.O.: Protein function in the post-genomic era. Nature 405(6788), 823–826 (2000)
8. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In: Fifteenth International Conference on Machine Learning. Morgan Kaufmann, San Francisco (1998)
9. Freitas, A.A.: Data Mining and Knowldge Discovery with Evolutionary Algorithms. Springer, Heidelberg (2002)
10. Furnkranz, J.: Separate-and-conquer rule learning. Artificial Intelligence Review 13(1), 3–54 (1999)
11. Galperin, M.Y., Koonin, E.V.: Whos your neighbor?New computational approaches for functional genomics. Nat. Biotechnol. 18, 609–613 (2000)
12. Gavin, A.C., Bsche, M., Krause, R., Grandi, P., Marzioch, M., Bauer, A., Schultz, J., Rick, J.M., Michon, A.M., Cruciat, C.M., Remor, M., Hfert, C., Schelder, M., Brajenovic, M., Ruffner, H., Merino, A., Klein, K., Hudak, M., Dickson, D., Rudi, T., Gnau, V., Bauch, A., Bastuck, S., Huhse, B., Leutwein, C., Heurtier, M.A., Copley, R.R., Edelmann, A., Querfurth, E., Rybin, V., Drewes, G., Raida, M., Bouwmeester, T., Bork, P., Seraphin, B., Kuster, B., Neubauer, G., Superti-Furga, G.: Functional organization of the yeast proteome by systematic analysis of protein complexes. Nature 415, 141–147 (2002)
13. Ge, H., Liu, Z., Church, G.M., Vidal, M.: Correlation between transcriptome and interactome mapping data from Saccharomyces Cerevisiae. Nat. Genet. 29, 482–486 (2001)

14. Giordana, A., Sale, C.: Learning structured concepts using genetic algorithms. In: Sleeman, D., Edwards, P. (eds.) Proceedings of the 9th International Workshop on Machine Learning, pp. 169–178 (1992)
15. Goh, C., Bogan, A.A., Joachimiak, M., Walther, D., Cohen, F.E.: Co-evolution of Proteins with their Interaction Partners. J. Mol. Biol. 299, 283–293 (2000)
16. Goh, C., Cohen, F.E.: Co-evolutionary Analysis Reveals Insights into ProteinProtein Interactions. J. Mol. Biol. 324, 177–192 (2002)
17. Ho, Y., Gruhler, A., Heilbut, A., Bader, G.D., Moore, L., Adams, S.L., Millar, A., Taylor, P., Bennett, K., Boutilier, K., Yang, L., Wolting, C., Donaldson, I., Schandorff, S., Shewnarane, J., Vo, M., Taggart, J., Goudreault, M., Muskat, B., Alfarano, C., Dewar, D., Lin, Z., Michalickova, K., Willems, A.R., Sassi, H., Nielsen, P.A., Rasmussen, K.J., Andersen, J.R., Johansen, L.E., Hansen, L.H., Jespersen, H., Podtelejnikov, A., Nielsen, E., Crawford, J., Poulsen, V., Srensen, B.D., Matthiesen, J., Hendrickson, R.C., Gleeson, F., Pawson, T., Moran, M.F., Durocher, D., Mann, M., Hogue, C.W., Figeys, D., Tyers, M.: Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry. Nature 415, 180–183 (2002)
18. Iqbal, M., Freitas, A.A., Johnson, C.G.: Protein Interaction Inference Using Particle Swarm Optimization Algorithm. In: Marchiori, E., Moore, J.H. (eds.) EvoBIO 2008. LNCS, vol. 4973, pp. 61–70. Springer, Heidelberg (2008)
19. Iqbal, M., Freitas, A.A., Johnson, C.G., Vergassola, M.: Message-Passing Algorithms for the Prediction of Protein Domain Interactions from Protein-Protein Interaction Data. Bioinformatics 24(18), 2064–2070 (2008)
20. Ito, T., Chiba, T., Ozawa, R., Yoshida, M., Hattori, M., Sakaki, Y.: A comprehensive two hybrid analysis to explore the yeast protein interactome. PNAS 98, 4569–4574 (2001)
21. Jansen, R., Yu, H., Greenbaum, D., Kluger, Y., Krogan, N.J., Chung, S., Emili, A., Snyder, M., Greenblatt, J.F., Gerstein, M.: A Bayesian Networks Approach for Predicting Protein-Protein Interactions from Genomic Data. Science 302, 449–453 (2003)
22. Lu, L.J., Xia, Y., Paccanaro, A., Yu, H., Gerstein, M.: Assessing the limits of genomic data integration for predicting protein networks. Genome Res. 15, 945–953 (2005)
23. Marcotte, E.M., Pellegrini, M., Ng, H.L., Rice, D.W., Yeates, T.O., Eisenberg, D.: Detecting protein function and protein-protein interactions from genome sequences. Science 285, 751–753 (1999)
24. Mewes, H.W., Frishman, D., Gldener, U., Mannhaupt, G., Mayer, K., Mokrejs, M., Morgenstern, B., Mnsterktter, M., Rudd, S., Weil, B.: MIPS:a database for genomes and protein sequences. Nucleic Acids Res. 30, 31–34 (2002)
25. Michalski, R.S.: On the quasi-minimal solution of the covering problem. In: Proceedings of the 5th International Symposium on Information Processing (FCIP 1969) (Switching Circuits), Bled, Yugoslavia, vol. A3, pp. 125–128 (1969)
26. Michalski, R.S.: AQVAL/1—Computer implementation of a variable-valued logic system VL_1 and examples of its application to pattern recognition. In: Proceedings of the First International Conference of Pattern Recognition, pp. 3–17 (1973)
27. Michalski, R.S., Mozetič, I., Hing, J., Lavrač, N.: The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In: Proceedings of the Fifth National Conference on Artificial Intelligence, pp. 1041–1045 (1986)
28. Oti, M., Snel, B., Huynen, M.A., Brunner, H.G.: Predicting disease genes using protein-protein interactions. J. Med. Genet. 43, 691–698 (2006)
29. Pagallo, G., Haussler, D.: Boolean feature discovery in empirical learning. Machine Learning 5, 71–99 (1990)
30. Quinlan, J.R.: Induction of decision trees. Machine Learning 1, 81–106 (1986)

31. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
32. Quinlan, J.R., Cameron-Jones, R.M.: Induction of logic programs: FOIL and related systems. New Generation Computing 13(3-4), 287–312 (1995)
33. Rhodes, D.R., Tomlins, S.A., Varambally, S., Mahavisno, V., Barrette, T., Kalyana-Sundaram, S., Ghosh, D., Pandey, A., Chinnaiyan, A.M.: Probabilistic model of the human protein-protein interaction network. Nature Biotechnology 23(8), 951–959 (2005)
34. Salwinski, L., Miller, C.S., Smith, A.J., Pettit, F.K., Bowie, J.U., Eisenberg, D.: The Database of Interacting Proteins: 2004 update. NAR 32, D449–D451 (2004)
35. Schaffer, C.: Overfitting avoidance as bias. Machine Learning 10, 145–154 (1993)
36. Shoemaker, B.A., Panchenko, A.R.: Deciphering ProteinProtein Interactions. Part-I: Experimental Techniques and Databases. PLoS Computational Biology 3(3), e42 (2007)
37. Shoemaker, B.A., Panchenko, A.R.: Deciphering ProteinProtein Interactions. Part-II: Computational Methods to Predict Protein and Domain Interaction Partners. PLoS Computational Biology 3(4), e43 (2007)
38. Thatcher, J.W., Shaw, J.M., Dickinson, W.J.: Marginal fitness contributions of non-essential genes in Yeast. PNAS 95, 253–257 (1998)
39. Uetz, P., Giot, L., Cagney, G., Mansfield, T.A., Judson, R.S., Knight, J.R., Lockshon, D., Narayan, V., Srinivasan, M., Pochart, P., Qureshi-Emili, A., Li, Y., Godwin, B., Conover, D., Kalbfleisch, T., Vijayadamodar, G., Yang, M., Johnston, M., Fields, S., Rothberg, J.M.: A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. Nature 403(1), 623–627 (2000)
40. Utgoff, P.E.: Shift of bias for inductive concept learning. In: Michalski, R., Carbonell, J., Mitchell, T. (eds.) Machine Learning: An Artificial Intelligence Approach, vol. II, pp. 107–148 (1986)
41. Valencia, A., Pazos, F.: Computational methods for the prediction of protein interactions. Current Opinion in Structural Biology 12, 368–373 (2002)
42. von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S.G., Fields, S., Bork, P.: Comparative assessment of large-scale data sets of protein-protein interactions. Nature 417(6887), 399–403 (2002)
43. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
44. Xenarios, I., Salwnski, L., Duan, X.J., Higney, P., Kim, S.M., Eisenberg, D.: DIP: The Database of Interacting Proteins. A research tool for studying cellular networks of protein interactions. NAR 30, 303–305 (2002)
45. Yamanishi, Y., Vert, J.P., Kanehisa, M.: Protein network inference from multiple genomic data: a supervised approach. Bioinformatics 20(suppl.1), i363–i370 (2004)
46. Yook, S.H., Oltvai, Z.N., Barabsi, A.L.: Functional and topological characterization of protein interaction networks. Proteomics 4, 928–942 (2004)
47. Yu, H., Greenbaum, D., Xin Lu, H., Zhu, X., Gerstein, M.: Genomic analysis of essentiality within protein networks. Trends Genet. 20, 227–231 (2004)

Improvements in Flock-Based Collaborative Clustering Algorithms

Esin Saka and Olfa Nasraoui

Abstract. Inspiration from nature has driven many creative solutions to challenging real life problems. Many optimization methods, in particular clustering algorithms, have been inspired by such natural phenomena as neural systems and networks, natural evolution, the immune system, and lately swarms and colonies. In this paper, we make a brief survey of swarm intelligence clustering algorithms and focus on the flocks of agents-based clustering and data visualization algorithm, (FClust). A few limitations of FClust are then discussed with proposed improvements. We thus propose the FClust-annealing algorithm that decreases the number of iterations needed to converge and improves the quality of resulting clusters. We also propose a (K-means+FClust) hybrid algorithm which decreases the complexity of FClust from quadratic to linear, with further improvements in the cluster quality. Experiments on both artificial and real data illustrate the workings of FClust and the advantages of our proposed variants.

1 Introduction

Clustering is the problem of finding groups in a dataset, according to some data properties and attributes which have a meaning in some context [14, 13]. Since no class information is used to cluster the data, it is called unsupervised learning. In addition to being an interesting and challenging problem, applications of clustering include customer segmentation in marketing, image segmentation, document organization, web usage mining, etc.

Esin Saka
Knowledge Discovery and Web Mining Lab, University of Louisville, KY, USA
e-mail: esin.saka@louisville.edu

Olfa Nasraoui
Knowledge Discovery and Web Mining Lab, University of Louisville, KY, USA
e-mail: olfa.nasraoui@louisville.edu

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 639–672.
springerlink.com

One of the many different approaches used for clustering is swarm intelligence (SI). *Swarm intelligence* is an artificial intelligence paradigm which is mainly inspired from the dynamics of several societies in nature, such as ant-colonies, bird-flocks, fish-schools, etc. SI is based on the social, collective and structured behavior of decentralized, self-organized agents [16, 36]. Although these agents have a very limited individual capacity, cooperatively they perform many complex tasks. Characteristics of swarm intelligence are: 1) Collaboration: agents in the swarm collaborate or interact with the environment and each other; 2) Collective intelligence: whereas agents in the swarm are mostly unintelligent, the collaborating system, or swarming mechanism results in an intelligent system; 3) Inspiration from nature; and 4) Decentralized control. In this chapter, we will mainly focus on using SI for clustering.

Given the above definition, the most popular swarm intelligence clustering algorithms are:

1. Ant-clustering
2. Particle swarm clustering
3. Flocks of agents-based clustering

There are mainly two approaches for ant-based clustering. In the first version, data is randomly placed on a grid. Then the ants move around the grid and form clusters by picking up and dropping the data items while moving [20]. Later, this version was improved in [34, 9, 10]. In the second version of the ant clustering algorithm, ANTCLUST, ants represent data items. Initially, none of the ants are assigned to a cluster, i.e. none of the ants have a label. During the clustering process, in each iteration, two randomly selected ants meet each other. Then, according to some defined *behavioral rules*, they may form a new cluster, one of the ants may be assigned to an existing cluster, one of the ants maybe removed from a cluster, or clustering quality measures may be updated [11, 17, 18, 19].

Clustering with particle swarms is based on particle swarm optimization[16, 15]. In the clustering problem, each particle encodes all cluster centroids. In other words, each particle represents a complete clustering solution [1, 22].

Lately, an approach based on flocks of agents, known as FClust [30, 29], was used for data clustering. This approach is inspired by bird flocks. Each agent of the flock represents a data item. Initially, agents are placed on a planar surface (hereinafter referred to as the *visualization panel*). Then, in each iteration, their speed gets updated according to the neighboring agents. In the end, similar agents start moving together and they form clusters. This makes FClust especially useful for *data visualization*. Although the experimental results given in [30] were acceptable, we observed that the standard deviations of the number of clusters, the cluster error, and the number of required iterations were rather high. Moreover, FClust was not successful for each data set. Another disadvantage was the high computational cost which can make FClust costly for many real time applications.

This chapter starts by reviewing algorithms for particle swarm clustering and ant clustering in Section 2. Section 3 reviews flocks of agents-based clustering, while pointing to their limitations. Then we will present several modifications in

Section 4, including an annealing variant, and a (K-means+FClust) Hybrid algorithm to overcome some of these limitations. In Section 6, post processing and experimental results are presented. Finally, our conclusion, in Section 7, summarizes the study and discusses future work.

2 Swarm Intelligence Clustering

Swarm intelligence is an artificial intelligence paradigm based on social, collective and structured behavior of decentralized, self-organized agents [16, 36]. Algorithms in this domain, mainly depend on an inspiration from nature, in particular from social insects like bees, ants, termites; flocks of birds, and fish schools. These animals have a very limited individual capacity. Yet, cooperatively, they perform many complex tasks such as searching for and storing food, and flying collectively over long distances. The characteristics of swarm intelligence are:

- *Collaboration :* Agents in the swarm collaborate or interact with the environment and with each other.
- *Collective intelligence:* Whereas agents in the swarm are mostly unintelligent, the collaborating system, or swarming mechanism results in an intelligent system.
- *Inspiration from nature:* Agents' properties tend to be derived from analogous creatures such as ants, bees, or birds.
- *Decentralized control:* Agents behave and interact without a centralization mechanism.

Five basic principles of swarm intelligence are [23]:

- **Proximity principle:** The group should be able to perform simple space and time computations.
- **Quality principle:** The group should be able to respond to quality factors in the environment.
- **Principle of diverse response:** The group should not allocate all of its resources along excessively narrow channels.
- **Principle of stability:** The group should not change its behavior with every change in the environment.
- **Principle of adaptability:** The group should change its behavior if it is beneficial.

In this chapter, we will mainly focus on using SI for clustering. Most common SI algorithms for clustering are:

1. Ant-clustering
2. Particle swarm clustering
3. Flocks of agents clustering

In the following subsections, general information about ant-clustering and particle swarm clustering is provided. Following this section, we will discuss clustering using flocks of agents in detail.

2.1 Particle Swarm Clustering

Clustering with particle swarms is based on particle swarm optimization[16, 15], which was initially inspired from bird flocks. Though the initial motive was modeling human social behavior, PSO later became a very popular search and optimization technique.

In PSO, the population is a group of particles and each particle is a candidate solution to the problem. Therefore in the swarm intelligence concept, with PSO, a swarm is a solution *set*. Each particle flies through a multi-dimensional problem space, and every position represents a different solution. After each move, the position is evaluated by a fitness function as shown in Line 7 of Algorithm 1.. The fitness function aims to evaluate the performance of each particle which is the closeness of the solution represented by the particle to the global optimum solution. The personal best solution which is the best position visited by the particle so far is calculated and kept (Line 8 of the Algorithm 1.). The particle's best position and the global best, which is the best solution found in the neighborhood, are used for computing the particle's new location (Line 10 of the Algorithm 1.). Depending on the definition of neighborhood, there exist two different versions of PSO: 1)*gbest:* the neighborhood is the entire swarm 2) *lbest:* a swarm is divided into overlapping neighborhoods of particles and the best particle is determined in the neighborhood.

The PSO Clustering algorithm was first applied in image clustering [27, 28], which took the number of clusters as input and used the gbest version of PSO. In the clustering problem, each particle is constructed from all cluster centroids. In other words, each particle represents a clustering solution [1, 22].

Algorithm 1. The PSO Clustering Algorithm

Input: Dataset and the number of clusters K.
Output: Clustered data.

```
 1: Initialize each particle to contain K random cluster centroids.
 2: for iteration=1 to max do
 3:     for all particle i do
 4:         for all data record x do
 5:             Calculate the Euclidean distance of x to all cluster centroids in i.
 6:             Assign x to the closest cluster.
 7:         Calculate the fitness of the particle.
 8:     Find the global best position (among all particles) and personal best position of each
        particle.
 9:     Update the velocity of each particle based on the global and personal best positions.
10:     Update the cluster centroids based on the particles' velocities.
```

A hybrid model of PSO clustering with the K-means clustering algorithm was presented in [22], where one of the particles was initialized with the result of K-means. Another (PSO+K-means) hybrid model was used for document clustering in [5, 7], where the results illustrated that the hybrid PSO algorithm can generate more compact results than K-means and PSO. A survey and a modified PSO-based clustering algorithm was presented in [2].

2.2 Ant Clustering

There are mainly two approaches for ant-based clustering. In the first version, *Ant Clustering Algorithm (ACA)*, data is randomly placed in the environment, which is generally a two-dimensional plane with a square grid. As shown in Algorithm 2., the ants move around the grid, via a random walk or jumping to form clusters by picking up, transporting, and dropping the data items while moving around [20]. The picking and dropping operations are influenced by the similarity of the surrounding data items and the density of the ant's local neighborhood $f(x_i)$. Generally, the size of the neighborhood is 3×3. The probability of picking up increases when the data item is surrounded by dissimilar data items or the density of the neighborhood is low. Similarly, the probability of dropping increases when similar data items are encountered. After ACA is stopped, a post-processing algorithm such as an agglomerative clustering algorithm is run for cluster retrieval [9]. More recently, performance analysis and strategies for increased robustness were presented [9, 10]. Improvements were also proposed by adding a progressive vision scheme and including pheromone on the grid cells [34]. A good review can be found in [11].

Algorithm 2. The Ant Clustering Algorithm (ACA)

Input: Dataset.
Output: Clustered data.

```
Randomly scatter data items on the grid.
Randomly scatter ants on the grid.
for iteration=1 to max do
    for all ant $a_i$ do
        if $a_i$ is unladen and $a_i$'s grid position is occupied by item $x_i$ then
            Calculate $f(x_i)$ and calculate $prob_{pick-up}(x_i)$ using $f(x_i)$.
            if $prob_{pick-up}(x_i) \geq$ random() then
                Let $a_i$ pick up item $x_i$.
        else if $a_i$ is carrying item $x_i$ and $a_i$'s grid position is empty then
            Calculate $f(x_i)$ and calculate $prob_{drop}(x_i)$ using $f(x_i)$.
            if $prob_{drop}(x_i) \geq$ random() then
                Let $a_i$ drop item $x_i$ to i's current grid position.
        Move $a_i$ to a randomly selected, neighboring, unoccupied grid position.
    iteration++
```

In the second version of the ant clustering algorithm, ANTCLUST, each ant represents a data item. Initially, none of the ants are assigned to a cluster, i.e. none of the ants have a label. Then, during the clustering process, in each iteration, two randomly selected ants meet each other. According to some defined *behavioral rules*, they may form a new cluster, one of the ants may be assigned to an existing cluster, one of the ants may be removed from a cluster, or clustering quality measures may be updated [17, 18, 19]. The basic idea is that agents who carry similar data items attract each other, while agents who carry dissimilar agents repel each other, which results in the formation of groups. A general outline of the ANTCLUST Algorithm is given in Algorithm 3. and behavioral rules are given below.

Ant Behavioral Rules in ANTCLUST:

1. **New nest creation:** If two ants without clusters meet each other and they are similar enough, they form a new cluster.
2. **Adding an unlabeled ant to an existing nest:** If a labeled ant a_i meets an unlabeled ant a_j, and if they are similar enough, a_j is labeled with the same label of a_i, i.e. added to the cluster of a_i.
3. **Positive meeting between two nest-mates:** If two ants a_i and a_j have the same labels and they are similar enough, then the quality measures are increased for these ants and the cluster they belong to.
4. **Negative meeting between two nest-mates:** If two ants a_i and a_j have the same labels and they are not similar enough, then the quality measures are decreased for these ants, and the ant with smaller quality measure is unlabeled.
5. **Meeting between two ants of different nests:** If two ants a_i and a_j have different labels but they are similar enough, then the quality measures are decreased for these ants and the cluster they belong to. Then, the ant belonging to the smaller nest is moved to the other ant's bigger nest.
6. **Default rule:** If none of the above applies, do nothing.

Algorithm 3. The Ant Clustering Algorithm ANTCLUST

Input: Dataset.
Output: Clustered data.

```
Map ants to data items and initialize ants' quality measures. Initially ants do not have any
labels, i.e. they do not belong to any cluster.
for iteration=1 to max do
    Randomly choose two ants and apply the behavioral rules above.
    iteration++
Delete nests that do not contain enough ants.
Reassign ants without labels to the most similar nests.
```

Another clustering algorithm called AntTree uses the ability of building mechanical structures of ants and builds a tree structure [3]. In this version, each data to be clustered represents a node of the tree and the algorithm searches for the optimal edges.

3 Flocks of Agents for Data Visualization and Clustering

A recent swarm intelligence approach uses a *flock of agents*. One of the definitions given for a *flock* is "*a number of animals of one kind, esp. sheep, goats, or birds, that keep or feed together or are herded together*"[1].

"The motion of a flock of birds is one of nature's delights" according to Craig Reynolds who has simulated this beauty in computer animation, where the bird-like, birdoid object is called *boid* [32]. One of the biggest differences between a particle and a boid in simulation is that boids have *orientation*, which makes them suitable for *data visualization* as well as clustering.

Studies about flocks of agents in computer science have mainly started with simulating moving bird flocks, based on two balanced and opposing behaviors of natural flocks, namely, 1) Desire to stay close to the flock, and 2) Desire to avoid collisions. These are simulated in the following three behaviors [32].

Natural Bird Flock Behaviors:

1. **Collision Avoidance/Separation:** Steering away from the other boids to avoid collision.
2. **Alignment/Velocity Matching:** Aiming to match the moving direction (i.e. heading) and speed to that of nearby flockmates.
3. **Cohesion/Flock Centering:** Attempting to adjust steering toward the average position of local flockmates and to stay close to the neighbors.

While cohesion and velocity matching represent the attraction forces, which keep the boids together, collision avoidance formed the rejection/repelling force. Other studies also tried to present behavioral rules and model collective behavior of animals [12, 4]. Later studies also focused on visualizing data using flocks of agents. Each individual boid represented one data item and a fourth behavior was added to represent moving with similar data items [31]:

4. **Information Flocking:** Attempting to move with similar boids.

The fourth behavior is pretty similar to the second behavior, velocity matching. However, in the fourth behavior, the aim is not moving together with all neighbors, but only with the ones similar enough to form a group. This behavior provided a suitable ground for using flocks of agents for data visualization and offered a motivation for data clustering.

It should be noted that, just as a flock can be formed of birds, it can also be formed by other boids such as fish, sheep, etc. Therefore, for the sake of generality, in this study, instead of the word boid, we use the word "agent".

The Multiple Species Flocking clustering model (MSF) [6] used flock clustering for data clustering and implemented a distributed multi-agent system. In that study, MSF was compared to Ant clustering and K-Means clustering algorithms. MSF converged faster than ant clustering. However, the clustering results for MSF

[1] http://dictionary.reference.com/browse/flock

were manually generated. The user looked at the visualization panel and selected the clusters. The results showed that MSF performed better than K-Means. Lately, in [30], a detailed flock clustering algorithm (FClust) was presented with a stopping criteria and automated cluster extraction algorithm. An application of this approach to Web usage mining can be found in [33]. In the following sections, we will describe the FClust algorithm, discuss its limitations, and suggest improvements with real life application examples.

3.1 Flocks of Agents Based-Data Visualization

Data visualization using flocks of agents is suitable for any kind of data set where one can define a similarity measure between data items. A flock consists of several agents, with each agent mapped to and representing one data record. As mentioned in Section 3, flocks are different from ordinary particles because they have *orientation*. Agents in a flock are attracted to similar agents and are repelled by the different agents. Moreover, the distance between the agents depends on the similarity between the data items that are mapped to those agents. Therefore, the visualization panel visualizes the similarity relation between the data items. Normally, data sets with at most three attributes can be visualized by a simple plot. However, when there are more than three attributes, this becomes harder. In particular, when there is a huge number of attributes, as in web usage data, data visualization becomes a challenging job.

When flocks of agents are compared to other swarm intelligence algorithms, such as ants and particle swarms, we find that flocks are more suitable for data visualization. In the case of ants, data items are moved on a rigidly structured *grid* by ants and placed on the same stack with similar items. However, distance does not necessarily represent the similarity, as in the case of flocks. The distance between two neighboring agents in a flock is inversely proportional to the similarity between them whereas the similarity between two ants only increases the chance of data items being neighbors, but does not define how close/far they are. The distance between two more similar data items may be bigger than the distance between two other items which are less similar. In the case of particle swarms, each particle does not represent one data item, but rather represents a clustering solution itself. Therefore, particle swarms may not be as suitable for data visualization, either.

It has been mentioned that neural networks can also be used for data visualization. However, most neural networks have a rather static structure whereas a flock of agents is inherently dynamic [31]. Also, neural networks are a centralized learning mechanism, whereas agent flocks are decentralized.

3.2 Flocks of Agents-Based Clustering

In the clustering with flocks of agents approach [30], each agent represents one data item. Initially, agents are placed on the *visualization panel*, which is a 2 or 3-dimensional continuous space, where x, y (and if applicable z) coordinate values range between 0 and 1. Agents may be placed randomly or some background information can be used to place them. Then, they start moving around. As they meet other agents in a defined neighborhood, they try to remain at an ideal distance from each other, which is determined according to the similarity of the original data items that agents are representing. The more the data items are similar, the smaller the ideal distance will be. Ideal distances are computed for each agent pair once at the beginning of the algorithm, based on the intrinsic properties or attributes of the data items. If neighboring agents are further apart than the ideal distance, there will be an attraction force between them and the agents will try to move closer to each other. In contrast, if the distance is less than the ideal distance, then there will be a rejection force, and agents will move apart from each other. Given this basic idea, Algorithm 4. gives the procedure for *Flocks of Agents Clustering (FClust)*.

Algorithm 4. FClust Algorithm [30]

Input: Dataset.

Output: Visualization of interaction between the data items. Agents corresponding to more similar items are located closer in the 2D visualization panel.

```
Initially place the agents on the visualization panel.
Initialize velocities of all agents.
Compute the ideal distances, $d_{ideal}$, between agents.
repeat
    for each agent i do
        for all j such that $d(j,i) \leq d_{th}$ and $i \neq j$ do
            if $d(i,j) = d_{ideal}(i,j)$ then
                $\beta(i,j) \leftarrow 0$
            else if $d(i,j) > d_{ideal}(i,j)$ then {attraction}
                $\beta(i,j) \leftarrow 4 \times \left(\frac{d(i,j)-d_{ideal}(i,j)}{d_{th}-d_{ideal}(i,j)}\right)^2$
            else {repulsion}
                $\beta(i,j) \leftarrow -4 \times \left(1 - \frac{d(i,j))}{d_{ideal}(i,j)}\right)^2$
            $\overline{v}_{resulting}(i,j) \leftarrow \overline{v}(j) + \beta(i,j) \times \overline{v}_{cap}(i,j)$
        if $\exists\, j$ such that $d(j,i) \leq d_{th}$ and $i \neq j$ then
            $\overline{w}(i)$ = normalize ( $\sum_{j|d(j,i)\leq d_{th} \& i\neq j} \overline{v}_{resulting}(i,j)$)
            if The angle between $\overline{v}(i)$ and $\overline{w}(i)$ is less than or equal to 90 degrees then
                $\overline{v}_{next}(i) \leftarrow \overline{w}(i)$
            else
                $\overline{v}_{next}(i) \leftarrow \overline{v}(i)$
        else
            $\overline{v}_{next}(i) \leftarrow \overline{v}(i)$
        $amp_{next}$(i) $\leftarrow$ $amp_{def}$ + $\frac{d_{th}}{20\times(neighbor\_no(i)+1)}$
    for each agent i do
        compute new position $p_{next}(i) \leftarrow p_{current}(i) + amp_{next}(i) \times \overline{v}_{next}(i)$
    Move all agents to the updated positions and update current velocities.
until Clusters are formed
```

In steps 1 and 2, the initialization is performed. The velocity vector $\overline{v}$, is a unit vector, (i.e. $||\overline{v}|| = 1$), representing the direction. In step 3, the ideal distances between agents are computed via Equation (1). Later, for each agent i, the neighboring agents that are close enough to i on the visualization panel, are extracted in Line 6, where *d(i,j)* is the 2D Euclidean distance between agents i and j. Then, for each neighbor:

- If the distance between the agents i and j is equal to the ideal distance between them (Line 7), there is no attempt to change i's velocity due to j (Line 8).
- If the distance between the agents i and j is greater than the ideal distance between them (Line 9), an attraction force will move i closer to j, with a more similar velocity to j (Line 10).
- If the distance between the agents i and j is smaller than the ideal distance between them (Line 11), a repelling force will move i further from j, with a less similar velocity to j (Line 12).

In line 13, the velocity effect on i due to neighbor j is computed where $\overline{v}_{cap}(i,j)$ is the unit vector pointing from i to j. Next is the computation of the updated velocity of agent i, $\overline{v}_{next}(i)$, between lines 14 and 21. First, if i has neighbors, then their resulting velocities on i are summed up and normalized. If the total, normalized velocity $\overline{w}$, does not change the agent's current direction more than 90 degrees, then the updated velocity is assigned as $\overline{w}$. Otherwise the velocity is kept unchanged for the next iteration. Similarly, if agent i does not have any neighbors -note that an agent is not a neighbor of itself- then the velocity will be kept the same for the next iteration. In line 22, the amplitude is computed depending on the number of neighbors and distance threshold, where amp_{def} is the default minimum amplitude. If the amplitude is too low, it may increase the number of iterations to converge. However, if the amplitude is too high, the agents may move further than the desired location. The minimum amplitude is empirically set to $\frac{1}{5} \times d_{th}$ [30]. At the end of each iteration, the updated agent coordinates are computed, and all the agents are moved to their updated positions simultaneously (lines 23 to 25). Moving agents around the visualization panel is performed until a stopping criteria is met and/or clusters are formed.

3.2.1 Setting the Parameters for FClust

The selection of parameters has a large impact on the convergence of the FClust algorithm. The first parameter is d_{th}, the distance threshold, which defines the neighborhood size (see line 6 in Algorithm 4.), and the latter affects the ideal distance via Equation (1) (see line 22 in Algorithm 4.) and amplitude. When d_{th} is too small, the agents cannot affect each other, and when it is too high, the algorithm does not converge. One method to compute the ideal distance between two agents, d_{ideal}, is given in Equation (1). If the ideal distances are overestimated, then the clusters cannot be observed on the visualization panel.

$$d_{ideal}(i,j) = \frac{1 - sim(i,j)}{1 - sim_{th}} \times d_{th}. \tag{1}$$

The similarity threshold, sim_{th}, in (1) is computed via Equation (2). If the similarity threshold is too large, then the algorithm will fail to converge, and if it is too small, then different clusters risk being combined into one cluster.

$$sim_{th} = \frac{sim_{average} + sim_{max}}{2} \tag{2}$$

3.2.2 Stopping Criteria for FClust

The most common method for stopping the algorithm is using human experts [31, 6, 30]. An expert keeps watching the visualization panel until stable clusters are formed. At that time, the algorithm is stopped. However, an automated method was also presented in [30]. The visualization panel, i.e. a 2D continuous space ([0,1]x[0,1]), was divided into 20 cells. For each cell, the spatial entropy which depends on the proportion p of agents located in this cell was computed, and if the observed minimum entropy has not improved for the last $3 \times n$ iterations, where n is the number of data records, the algorithm is stopped. The spatial entropy is given in Equation (3).

$$ES = -\sum_{i=0}^{20} p(i) \times ln(p(i)) \tag{3}$$

The problem with this criterion is that the entropy may remain unchanged for a while if new neighbors do not meet, but after a meeting occurs, changes may start re-occurring. Thus, the above stopping criterion cannot capture these delayed dynamics, and thus a risk that the algorithm will be stopped before convergence.

3.2.3 Cluster Formation in FClust

When FClust is run, flocks of agents are visually observable. However, the clusters are not explicitly formed and the data is not yet assigned to clusters. Similar to the stopping criterion, one method of forming clusters is using human experts. The person marks the clusters and assigns agents to the clusters. Since there is a one-to-one mapping between agents and data records, the data will also end up being clustered. In addition to this, an automated procedure was presented in [30], which is given in Algorithm 5. Basically, a new cluster is created for an unlabeled agent. Then the neighboring agents of this cluster are explored, and all the agents which are similar to at least one of the agents in the cluster are inserted into this cluster. New agents are inserted to the cluster until no more agents can be inserted. Then the procedure restarts by creating another new cluster, and stops when all the agents are labeled.

After cluster formation, a post processing phase is needed to cluster the original (input) data, to validate the results, and if possible to interpret the clusters.

Algorithm 5. Cluster Formation Algorithm

Input: Agents' coordinates at a stable/converged state.
Output: Clusters of agents.

```
for each agent $i$ do
    if $i$'s cluster is not assigned then
        Form a new cluster $c$
        Assign $i$ to $c$
        for all agent $j$ such that there exists an agent $k$ such that $k \in c$ and distance(k,j)$\leq$ $d_{th}$ and $sim(k,j) > sim_{th}$ do
            Assign $j$ to $c$
```

3.2.4 Complexity Analysis of FClust

The FClust algorithm, given in Algorithm 4., needs to compare every agent to every other agent in order to compute the ideal distance initially, and then to update the agent's velocity based on its neighboring agents. Although some complexity reduction suggestions, such as using a neighborhood matrix, was given [30], the worst case time complexity remains $O(n^2)$, where n is the number of data records. Similarly, the memory complexity is also $O(n^2)$ to keep the ideal distances, in addition to $O(n)$ memory needed for keeping agent locations, velocities and amplitudes.

3.2.5 Limitations of FClust

Although the experimental results given in [30] were acceptable, we observed that the standard deviations of the number of clusters, the cluster error, and the number of required iterations were high. Moreover, FClust was not successful for each data set. Another disadvantage was the high computational cost which makes FClust unsuitable for many real time applications.

In addition to the above limitations, we observed that convergence strongly depends on the similarity threshold. When the similarity threshold is too high, the algorithm may not converge, and when the threshold is too low, the algorithm may not differentiate between different clusters, and thus end up combining some of them. Therefore, Equation (2) is not suitable for every dataset. Furthermore, if the data similarity values follow a power low distribution, then the similarity threshold given in Equation (2) will produce a very high similarity threshold. Therefore agents will not be able to form clusters. In other words, the clustering algorithm will not converge. Another problem occurs when there are connecting agents between clusters, meaning that instead of being well separated, a bridge of data points connects two clusters. As a result the clusters are labeled as the same, even though they should be labeled differently. To solve this problem to some extent, an alternative formulation will be presented in Section 4. Moreover, the ideal distance formula in [30] requires mostly unique data records. Otherwise, if many similarity values are 1, the ideal distance computation results in an infinite value in Equation 1 because $sim_{th} = 1$. An alternative formulation, which can handle many 1-similarities is given in Section 4.1.

4 Improved Distance Threshold Estimates

4.1 Alternative Fixed Thresholding

In our experiments, we observed that the original FClust was not suitable for some very high dimensionality sparse datasets, such as Web usage data. Therefore, we had to find a new formulation for the similarity threshold as given by Equation (4). When the maximum similarity is very high and the average similarity very low, the similarity threshold computed by (2) is too high for FClust to converge. Equation (4) includes a flexible way to tune the similarity threshold, by a tuning factor α, where $\frac{sim_{max}}{sim_{average}} \geq \alpha \geq 1$.

$$sim_{th} = \alpha \times sim_{average} \tag{4}$$

Another problem observed is that the ideal distance computation results in an infinite value via Equation (1) if many similarity values are 1, because $sim_{th} = 1$. Equation 5 proposes an alternative formulation, which can handle many 1-similarities.

$$d_{ideal}(i,j) = \begin{cases} \frac{1-sim(i,j)}{1-sim_{th}} \times d_{th}, & sim_{th} \neq 1 \\ 0, & sim_{th} = 1 \end{cases} \tag{5}$$

Additionally, when the number of data records is small, setting the distance threshold is very hard because, if it is too low, the agents cannot meet each other, while if it is too high, the ideal distances computed via Equations (1) and (5) will be too high, and cluster formation will not occur. As a solution, a new parameter for ideal distance tuning is added, as given in Equation (6). The ideal distance threshold constant, d_{ideal_th}, used in Equation (6), is also used, replacing d_{th} during the cluster formation given in Algorithm 5.1, line 5.

$$d_{ideal}(i,j) = \begin{cases} \frac{1-sim(i,j)}{1-sim_{th}} \times d_{ideal\_th}, & sim_{th} \neq 1 \\ 0, & sim_{th} = 1 \end{cases} \tag{6}$$

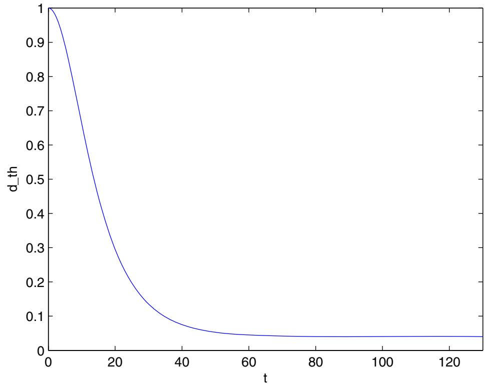
In addition to these simple tuning modifications, an annealing version of FClust is proposed in Section 4.2.

4.2 Adaptive Thresholding Using FClust-Annealing

The distance threshold, d_{th}, has a great effect on the convergence of clustering. When d_{th} is too high, too many other agents affect a given agent. This results in an attempt to satisfy too many constraints at once, and the algorithm does not converge. However, when d_{th} is too low, the neighborhood tends to be too narrow for agents to see each other. Therefore they may not affect each other. Even when convergence is possible, it takes too many iterations.

One way to decrease the sensitivity to a fixed threshold is by using an annealing schedule for d_{th}, where d_{th} is highest at the beginning and then decreases as the

Fig. 1 Cooling Schedule for d_{th} for Annealing FClust



number of iterations increases. In that case, d_{ideal} is computed using Equation (6). A possible cooling schedule for d_{th} is given in Equation (7) and shown in Figure 1.

$$d_{th}(t) = \frac{d_{th}(0) - d_{th}(N)}{cosh(\frac{10 \times t}{N})} + d_{th}(N) \tag{7}$$

where, t is the iteration number and N is the number of iterations. d_{th} starts from $d_{th}(0)$ and decreases down to $d_{th}(N)$.

During the cluster formation process, in line 5 of Algorithm 5., d_{ideal_th} is used to define the neighborhood.

5 The (K-means/FClust) Hybrid Algorithm

K-means is a fast algorithm with *O(n)* time complexity. However, the number of expected clusters, K, needs to be given as an input to the K-means algorithm and K-means may not cluster the data successfully if the cluster boundaries are not hyperspherical. Unlike K-means, FClust has an adaptive way to extract a reasonable number of clusters without any boundary restrictions. However, the complexity cost for FClust is higher than K-means. In this section, we propose a (K-means+FClust) Hybrid Algorithm which aims to benefit from the advantages of both K-means and FClust.

5.1 K-Means Algorithm

The K-Means Algorithm is a popular, simple, unsupervised, learning algorithm for clustering. [21]. In this iterative, partitional clustering approach, each cluster is associated with a cluster centroid and each data point is assigned to the cluster with the closest centroid.

The K-Means Algorithm, listed in Algorithm 6., aims to minimize an objective function, consisting of the sum of squared errors or distances between the data records and K cluster centroids, as given in Equation 8, by iteratively updating the cluster centroids and assigning data to the most similar centroid, until the total error between the data records and the assigned cluster centroids converges. Note that this procedure forms clusters with n-dimensional hyper-spherical boundaries, where cluster centroids represent the centers.

$$E = \sum_{j=1}^{d} \sum_{i=1}^{n} ||\chi_i^j - c_{\pi(\chi_i)}^j|| \tag{8}$$

where n is the number of data records, d is the number of attributes and $\pi(\chi_i) = \mathit{cluster\ to\ which\ } \chi_i \mathit{\ is\ assigned}$.

The complexity is $O(n * K * I * d)$ where n is the number of points, K is the number of clusters, I is the number of iterations, and d is the number of attributes.

Algorithm 6. K-Means Algorithm

Input: Dataset $\chi \in \Re^d$ where $|\chi|$=n; number of clusters, $K \leq \sqrt{n}$.
Output: A partition of the dataset into K disjoint clusters $\gamma_1, ..., \gamma_K$.

```
Read sessions.
Arbitrarily select $K$ records as centroids out of the $n$ data records.
repeat
    for all Data record $\chi_i$ do
        Find the closest centroid $c_i$ using Euclidean Distance.
        Assign data record $\chi_i$ to the cluster $\gamma_i$.
    for all Cluster $\gamma_j$ do
        Update its centroid $c_j = \frac{\sum_{\chi_i \in \gamma_i} \chi_i}{||\sum_{\chi_i \in \gamma_i} \chi_i||}$
until stopping criterion is met.
```

5.2 (K-means+FClust) Hybrid

K-means is a fast algorithm with *O(n)* time complexity. However, the number of expected clusters, K, needs to be given as an input to the K-means algorithm. Moreover, K-means can only find clusters with hyperspherical shapes. Therefore, if the clusters are not hyperspherical, K-means may not estimate the cluster boundaries correctly. A hybrid algorithm of K-means and FClust is presented in Algorithm 7. to take advantage of the speed of K-means while also benefiting from the power of automatically determining the number of clusters in FClust. In the hybrid algorithm, initially, K-means is run with a high number of clusters which is more than the expected number of clusters. The cluster centroids are then extracted to get cluster representatives, and these representatives are mapped to the agent domain. Next, FClust is run on this smaller number of agents (relative to the size of the input data set) and the clustering results are mapped back to the input data domain. Note that, in the hybrid approach, each agent is mapped to the closest group of data records via its group centroid, since each agent represents a group centroid. The time complexity for K-means, lines 1 to 2 in Algorithm 7., is *O(n)*, and the time complexity of the FClust part, in lines 3 to 4, is *O*(K^2). Since the number of agents K is very small compared to the number of agents n, as long as $K \leq \sqrt{n}$ (which is always the case in practice), the time complexity and memory complexity of the (K-means+FClust) hybrid is *O(n)*. Therefore the hybrid version reduces the *time and memory* complexities *from quadratic to linear*.

Algorithm 7. (K-means+FClust) Hybrid Algorithm

Input: Dataset with n data records, an over-specified number of initial clusters, $K \leq \sqrt{n}$.
Output: Visualization of interaction between the data items and agent-cluster formation.

```
1: Run K-means on the original dataset with $K$ clusters, where $K \leq \sqrt{n}$).
2: Extract $K$ cluster centroids from K-means' results,
3: Map each cluster centroid to an agent for FClust,
4: Run FClust with $K$ agents.
```

5.3 Advantages of the (K-means+FClust) Hybrid Algorithm

The advantages of the (K-means+FClust) Hybrid algorithm can be listed as:

- Linear complexity: Unlike FClust, the (FClust+K-means) Hybrid algorithm has linear complexity inherited from K-means.
- Unlike K-means, the (FClust+K-means) Hybrid algorithm has an adaptive way to extract a reasonable number of clusters.
- Unlike K-means, the (FClust+K-means) Hybrid algorithm suffers from no hyperspherical boundary restrictions on the extracted clusters.

5.4 Stopping Criterion

During the movement of the agents, the goal of each agent is to move such that the agent will be located at an ideal distance to all neighboring agents. Therefore, the procedure can be considered as a constraint satisfaction problem [35], where having the ideal distance between every agent pair represents the optimal goal. The difference between the real distance between two agents and their ideal distance is called the *ideal distance error* in this paper as shown in Equation (9). One proposed stopping criterion is to halt if the difference in total ideal distances between all agent pairs in two consecutive iterations is small enough. Since this stopping criterion is feasible when the number of agents is small, it will be used for the (K-means+FClust) Hybrid algorithm in the experiments of Section 6.3.

$$\mathit{ideal\ distance\ error}(i,j) = |d_{ideal}(i,j) - d(i,j)| \tag{9}$$

6 Experimental Results

In this section, we describe our experiments and their results. We start by describing the datasets that we used in our experiments, in Section 6.1. Then, we proceed to explaining how post processing is applied to extract the clusters in Section 6.2. Finally, Section 6.3 presents the experimental results observed for FClust, FClust-annealing, and (K-means+FClust) Hybrid on different datasets. No experiments are presented for (K-means+FClust) Hybrid-Annealing because, the aim behind annealing is to initially have a bigger neighborhood and then reduce it with time to speed up the convergence of clusters. However, in the hybrid version, since the number of agents is already small, a bigger distance threshold, thus a wider neighborhood size, is being used, and naturally convergence is very fast. Therefore, annealing is not used for the hybrid experiments.

6.1 Datasets

As shown in Table 1, datasets I and II are synthetic datasets, whereas dataset WebM consists of real Web usage sessions. Iris and Pima are also real life datasets from the UCI machine learning repository [2]. Datasets I and II have 2 attributes and are thus suitable to show the clustering results visually. Dataset WebM, the Web usage data, consists of Web usage *sessions*, where each session is a bag of visited *URLs*. Each session includes the URLs that were visited during that session. In Web usage mining, each URL or item is considered as one dimension which results in a huge dimensionality. To compare the proposed improvements and the hybrid algorithm with the original FClust algorithm, datasets Iris and Pima are also used. In the experiments, two different similarity measures are used, the Manhattan based similarity for datasets I and II, and the cosine similarity for dataset WebM. The Manhattan based similarity is used for the linearly normalized Iris and Pima datasets.

The Manhattan based (L1) similarity of two data records x_i and x_j is given by

$$sim(x_i, x_j) = 1 - \frac{1}{A} \sum_{k=1}^{A} |x_i^k - x_j^k|, \qquad (10)$$

Table 1 Datasets

Dataset ID	**Number of Items**	**Number of Attributes**	**Number of Clusters**	**Maximum Similarity**	**Average Similarity**
I	1600	2	2	0.999948	0.814345
II	811	2	2	0.997500	0.732619
WebM	1704 (sessions)	343 (urls)	NA	1.000000	0.059718
Iris	150	4	3	1.0	0.709334
Pima	768	8	2	0.986171	0.832665

[2] http://archive.ics.uci.edu/ml/

where A denotes the number of attributes and x_i^k denotes the k^{th} attribute of data record x_i. When the data is linearly normalized to $[0,1]$, the Manhattan based similarity is the same as the 1-norm similarity, which is used in (10).

Given that $\overline{s_i}$ and $\overline{s_j}$ are two sessions, each formed of a list of $|\overline{s_i}|$ and $|\overline{s_j}|$ URLs visited in each user session respectively, the cosine similarity is computed as follows:

$$sim(\overline{s_i},\overline{s_j}) = \frac{|\overline{s_i}\bigcap\overline{s_j}|}{\sqrt{|\overline{s_i}|\times|\overline{s_j}|}} \tag{11}$$

6.2 *Post Processing*

After the agent clusters are formed, a post-processing phase is needed to cluster the data and validate the results. Post-processing depends on the data properties. If the data has 3 attributes or less, the data points are plotted with different colors depending on the cluster assigned. If the data has a class attribute, then the cluster error, given in Algorithm 11, is also computed.

Algorithm 8. Synthetic Data Post-Processing Algorithm

Input: Agents' coordinates and their cluster labels, Original data set, S_{TH}: minimum cluster size.

Output: Clustered data, plot of agents and data set colored according to their cluster label.

```
Create as many clusters as formed for the agents.
Label each data record with the same label as the agent representing that data record.
Keep only the clusters with enough data records (i.e. size is above $S_{TH}$).
Plot agents on the visualization panel, colored according to cluster label.
If the data is in 2D or 3D, then plot the data records, colored according to the cluster label
(for validation).
If the data has a class label, compute the cluster error via Algorithm 11.
```

Algorithm 9. Web Usage Data Post-Processing Algorithm

Input: Agents' coordinates and their cluster labels, Original data set, S_{TH}: minimum cluster size.

Output: Clustered data, plot of agents colored according to their cluster label, and user profiles.

```
Create as many clusters as formed for the agents.
Label each session with the same label as the agent representing that session.
Keep the clusters with more than $S_{TH}$ sessions.
Plot agents on the visualization panel, colored according to their cluster label.
for all Clusters do
    Find the URLs which are visited more than $item\_count\_threshold$ in all the sessions
    of that cluster.
```

Algorithm 10. (K-Means+FClust) Post-Processing Algorithm

Input: Cluster means produced by K-means, agents' coordinates and cluster labels from FClust's output.

Output: Clustered data, and plot of agents and dataset, colored according to their cluster label.

```
1: Create as many clusters as formed for the agents
2: Map data records to the cluster means of K-means, in other word to agents.
3: If data is in 2D or 3D, plot data records, colored according to their cluster labels from
   K-means.
4: Label each data record with the same label as the FClust-generated label of the agent
   representing that data record.
5: Keep the clusters with more data records than $S_{TH}$.
6: Plot agents on the visualization panel, colored according to their cluster label.
7: If data is in 2D or 3D, plot data records, colored according to their cluster label from
   FClust.
8: If the data has a class label, compute the cluster error via Algorithm 11.
```

Algorithm 11. Cluster Error Computation Algorithm

Input: Dataset with class labels and cluster labels from FClust output.

Output: Cluster error -a real number between 0 and 1-.

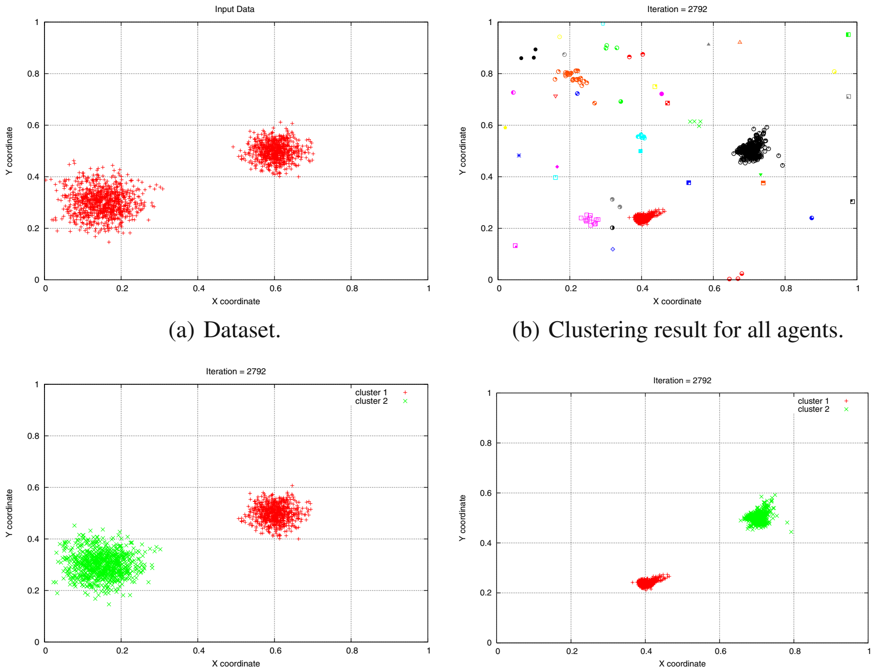
```
$Error \leftarrow 0$
for all Data record pairs $(i, j)$, where $i \neq j$ do
    if $i$ and $j$ have the same class label, but different cluster labels then
        $Error++$
    else if $i$ and $j$ have different class labels, but the same cluster label then
        $Error++$
Return $Error/Number\ of\ pairs$
```

Moreover, if the data consists of Web user sessions, then profiles are extracted as shown in Algorithm 9., line 6. In Algorithm 8., line 3, Algorithm 9., line 3, and Algorithm 10., line 5, S_{TH}, denotes the *session threshold*, i.e. the minimum number of sessions required for a cluster to be valid. If a data record is a bag of items, as in the case of Web usage data, the value of item_count_threshold used in Algorithm 9., line 6 is given in Equation (12), where $ICTF$ denotes the *Item Count Threshold Frequency*, where $0 \leq ICTF \leq 1$ is a real number. As a result, each set of URLs, extracted in line 6 of Algorithm 9., can be considered as a pattern that represents a Web user profile that summarizes the sessions assigned to that cluster.

$$item\_count\_threshold = ICTF * cluster\_size \tag{12}$$

6.3 Results

In Sections 6.3, we start with 2D datasets to allow us to do a visual evaluation. Then we proceed with the Web usage data as a challenging, high dimensional, real



(a) Dataset.

(b) Clustering result for all agents.

(c) Clustering result for dataset.

(d) Agent clusters generated after post processing and assigning agents to clusters.

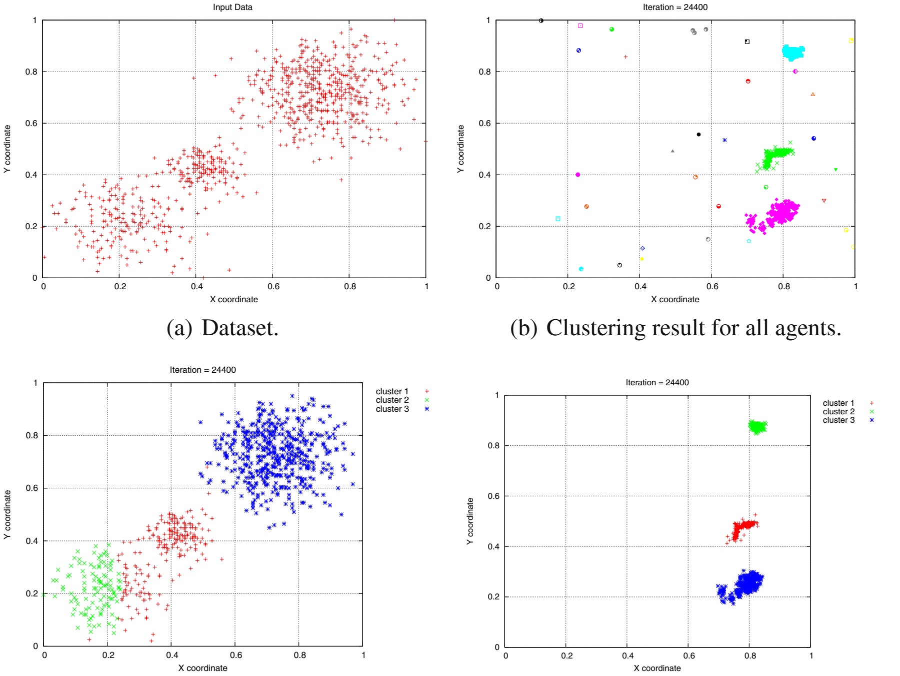
Fig. 2 Clustering a dataset with two clusters using FClust where d_{th}=0.04, sim_{th}=0.91

life data example. Finally, we present our results for the Iris and Pima datasets and compare the clustering outputs to the data class provided as part of the datasets.

6.3.1 FClust Results for 2D and Web Usage Datasets

Figure 2(a) shows an example of a data set with 2 clusters, and the resulting agents space are shown in Figure 2(b). In Figure 2(d), agent clusters which include more than S_{TH} data records are shown. And in Figure 2(c), the clustered data is shown. Figure 3 shows the result for a more complicated data set, given in Figure 3(a), using the Manhattan based (i.e. L1) similarity given in Equation (10). Likewise, Figure 3(d) is the post-processed version of Figure 3(b). When we compare the results in Figure 2(c) and Figure 3(c), we observe that, when the data clusters are not strictly separated, the cluster formation algorithm, Algorithm 5, may suffer from a bridging effect that results in merging two distinct clusters. Note that, since the synthetic datasets used in our experiments already had attributes between 0 and 1, we did not linearly normalize them in [0,1].

Figure 4 shows the results after more iterations compared to Figure 3. This shows that, if the agents' movement is stopped in a wrong state, different clusters may be



(a) Dataset.

(b) Clustering result for all agents.

(c) Clustering result for dataset.

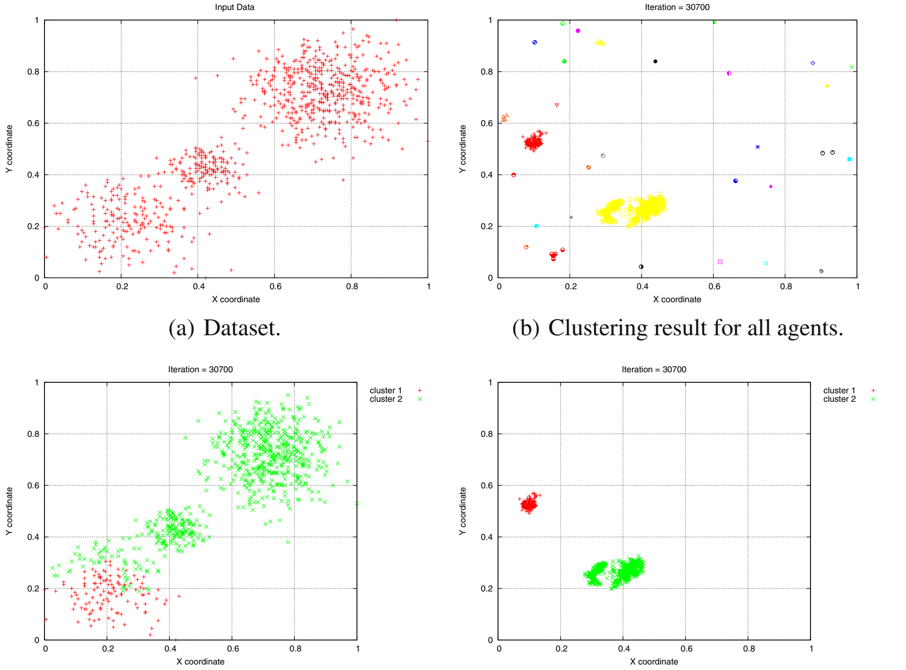
(d) Agent clusters generated after post processing and assigning agents to clusters.

Fig. 3 Clustering a dataset with three clusters using FClust at iteration 24400 where d_{th}=0.04, sim_{th}=0.86

assigned to the same cluster. Therefore, the stopping criteria is crucially important for overlapping data sets.

The next results are for two weeks worth of Web usage data for a Computer Engineering and Computer Science department's website. We have chosen this data set, because it has have previously undergone extensive experiments and validation in [26, 25, 24], hence it is considered a benchmark data set. In Figure 5(a), the algorithm did not converge because the similarity threshold computed according to Equation (2) was too high to form good clusters. With the average similarity and maximum similarity given in Table 1, the similarity threshold is computed as 0.53, which is very high given that the average similarity is 0.06. Therefore this example visually shows that the similarity threshold given in Equation (2) is not suitable for data with similarities distributed as a power law, as can be verified in Figure 5(b) and Figure 5(c) (the log-log plot exhibiting several linear segments).

If we compute the similarity threshold using (4), it is 0.15. The value of α was set to 2.5 in this set of experiments. We obtain convergence as shown in Figure 6, where S_{TH} = 10 and ICTF = 0.10. Several examples of the extracted profiles are presented



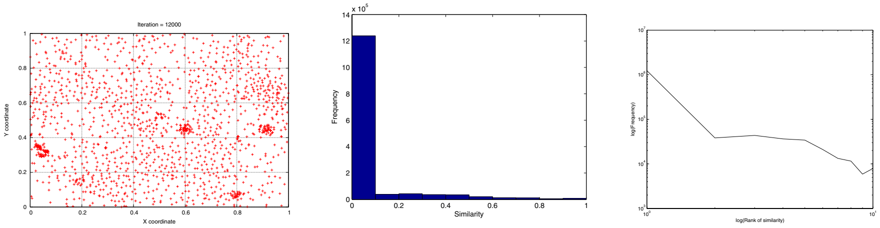
(a) Dataset.

(b) Clustering result for all agents.

(c) Clustering result for dataset.

(d) Agent clusters generated after post processing and assigning agents to clusters.

Fig. 4 Clustering a dataset with three clusters using FClust at iteration 30700 where d_{th}=0.04, sim_{th}=0.86



(a) Agents for Web usage data,

(b) Cosine similarity histogram,

(c) Log-log plot of similarities.

Fig. 5 Clustering the Web usage data using FClust with similarity threshold computed according to Equation (2). (a) no convergence because of an improper similarity threshold (b) Similarity histograms, (c) Log-log plot of similarities exhibiting power law properties

in Table 2. For example, *Profile*1 represents a group of users (possibly prospective students) checking the department's main web pages. *Profile*2 represents a group of student users taking the course CECS 352 (*Joshi* is the instructor teaching the course).

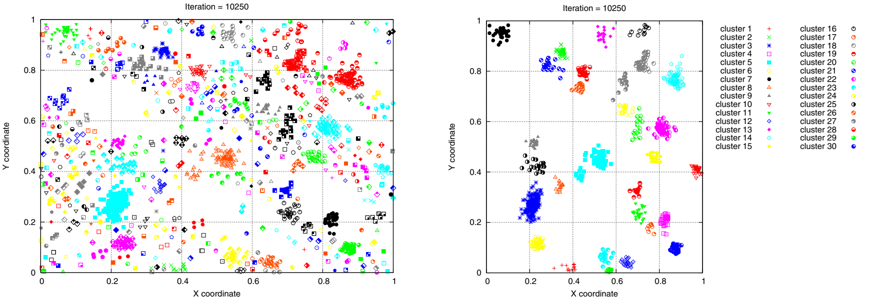
Table 2 Several examples from the profiles of a Web usage data extracted using FClust where $S_{TH} = 10$ and $ICTF = 0.1$

URL Frequency	URL
Profile 1 (includes 211 sessions)	
0.92	/
0.79	/cecs_computer.class
0.53	/courses.html
0.52	/courses_index.html
0.51	/courses100.html
0.27	/people.html
0.27	/people_index.html
0.27	/faculty.html
0.20	/courses300.html
0.19	/degrees.html
0.15	/courses200.html
0.13	/grad_people.html
0.12	/research.html
0.11	/courses_webpg.html
0.11	/index.html
0.10	/staff.htm
Profile 2 (Includes 43 sessions)	
0.93	/ joshi/courses/cecs352
0.33	/ joshi/courses/cecs352/slides-index.html
0.26	/ joshi/courses/cecs352/outline.html
0.23	/ joshi/courses/cecs352/text.html
0.23	/ joshi/courses/cecs352/handout.html
0.21	/ joshi/courses/cecs352/proj
0.19	/ joshi
0.16	/ joshi/courses/cecs352/proj/proj1.html
0.14	/ joshi/courses/cecs352/proj/overview.html
0.12	/ joshi/courses/cecs438
0.12	/ joshi/courses/cecs352/environment.htm

6.3.2 FClust-Annealing Results for 2D and Web Usage Datasets

Figure 7 shows the clustering results for the dataset with 2 clusters using FClust-annealing. Comparing this result with Figure 2, we can see that the annealing results in fewer iterations to convergence (1410 vs. 2792 iterations).

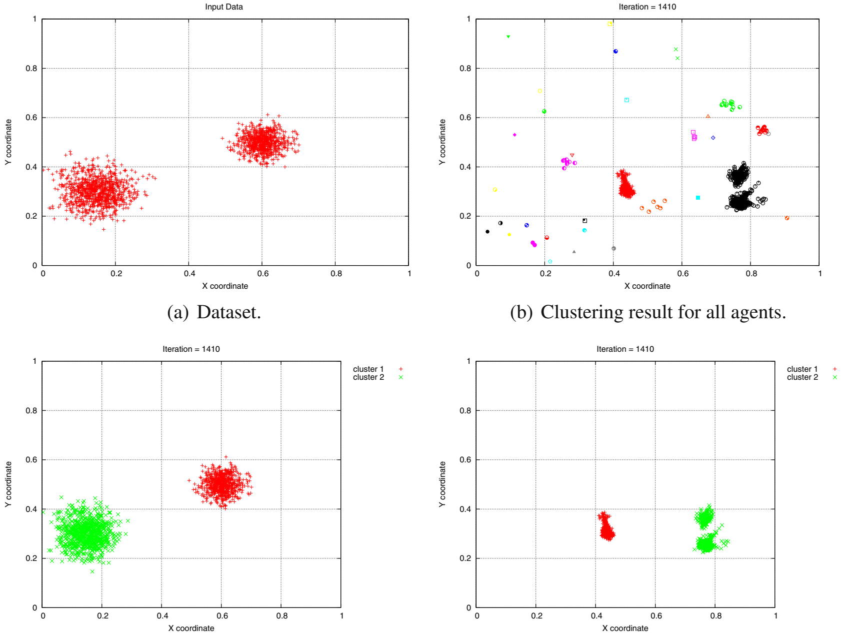
Figure 8 also shows that annealing not only accelerates the convergence, but also results in better quality clusters (no bridging effect). In Figure 3, even though there were 3 separate clusters in the agent space, after 24,400 iterations, as seen in Figures 3(b) and 3(d), clustering the data domain as shown in Figure 3(c) showed that the clustering process did not converge. Figure 4 also confirms the fact that there is a tendency to combine 3 clusters into one cluster unless a better cluster formation algorithm is applied. Although the classical FClust suffers from these problems, FClust-annealing clearly differentiates between these clusters. In addition to



(a) Clustered agents before pruning.

(b) After post-processing with Algorithm 9.

Fig. 6 Generated profiles from web usage data using FClust in 10250 iterations where d_{th}=0.04, sim_{th}=0.15, S_{TH}=10, ICTF=0.10



(a) Dataset.

(b) Clustering result for all agents.

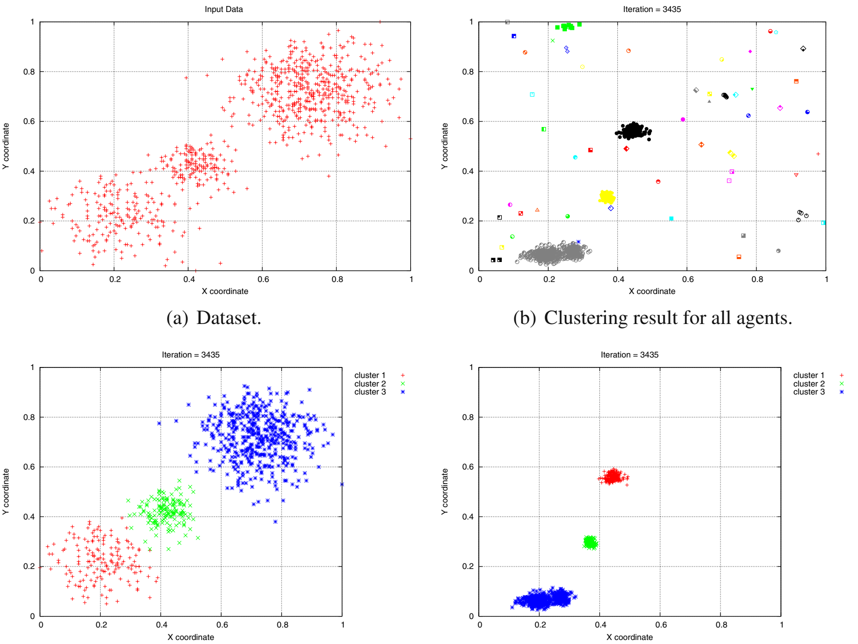
(c) Agent clusters generated after post processing and assigning agents to clusters.

(d) Agent clusters generated after post processing and assigning agents to clusters.

Fig. 7 Clustering a dataset with two clusters using FClust-annealing where d_{th}=started from 1 down to 0.04, sim_{th}=0.91

being more successful in clustering, FClust-annealing converges in fewer iterations (3,435), compared to 24,400 iterations for FClust, i.e. it was 7 times faster.

Figure 9 shows the results of clustering the real life Web usage data, listed in Table 1, using FClust with the annealing. Example profiles are shown in Table 3.



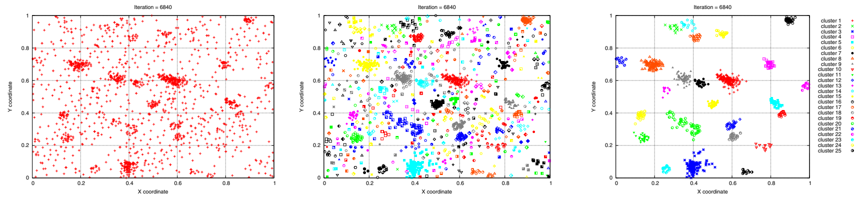
(a) Dataset.

(b) Clustering result for all agents.

(c) Clustering result for dataset.

(d) Agent clusters generated after post processing and assigning agents to clusters.

Fig. 8 Clustering a dataset with three clusters using FClust-annealing where d_{th}=started from 1 down to 0.04, sim_{th}=0.91, d_{ideal_th} for FClust and post-processing is 0.04



(a) Clustering result for all agents.

(b) Clustered agents before pruning.

(c) Agent clusters generated after post-processing and assigning agents to clusters.

Fig. 9 Clustering the Web usage session data using FClust-annealing where d_{th}=started from 1 down to 0.04, sim_{th}=0.15 , d_{ideal_th} for FClust and post-processing is 0.04

Compared to Figure 6 which took 10,250 iterations, better quality clusters are now formed in only 6,840 iterations. Also 30 clusters were formed in Figure 6 whereas, with annealing, 25 clusters are formed. By checking the post-processed profiles, we observed that the decrease in the number is not a loss of information but rather a better convergence (broken clusters were combined). In FClust, some clusters were

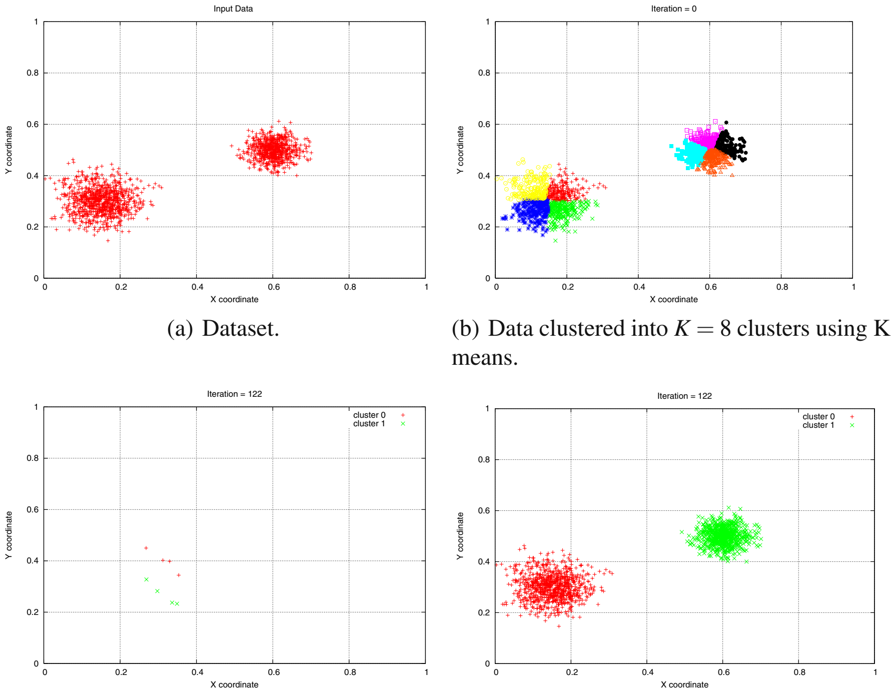
Table 3 Some samples from the Web user profiles, extracted using FClust-annealing where $S_{TH} = 10$ and $ICTF = 0.1$

URL Frequency	**URL**
Profile 1 (includes 116 sessions)	
0.90	/
0.69	/cecs_computer.class
0.43	/courses_index.html
0.42	/courses100.html
0.41	/courses.html
0.29	/people.html
0.28	/people_index.html
0.28	/faculty.html
0.20	/courses300.html
0.18	/degrees.html
0.17	/courses200.html
0.13	/general.html
0.13	/general_index.html
0.13	/facts.html
0.13	/research.html
0.11	/grad_people.html
Profile 2 (Includes 31 sessions)	
0.90	/ joshi/courses/cecs352
0.35	/ joshi/courses/cecs352/slides-index.html
0.35	/ joshi/courses/cecs352/handout.html
0.35	/ joshi/courses/cecs352/outline.html
0.29	/ joshi/courses/cecs352/text.html
0.26	/ joshi/courses/cecs352/environment.html
0.13	/ joshi
0.13	/
0.13	/ joshi/courses/cecs438
0.13	/ joshi/courses/cecs352/proj

broken and their agents could not meet each other on the agents visualization panel and therefore could not be unified.

6.3.3 (K-means+FClust) Hybrid Results for 2D Datasets

Figure 10 shows the results of the (K-means+FClust) Hybrid algorithm for Dataset I. The disadvantage of K-means is that it requires the number of clusters as input, however FClust can extract the number of clusters automatically. With the hybrid approach, 8 clusters are generated with K-means, as shown in Figure 10(b), where each agent was mapped to one cluster center generated by K-means. From these, FClust generated the 2 clusters, shown in Figure 10(c). Figure 10(b) represents the K-means result, (i.e. before starting the iterations of FClust). Compared to Figure 2, where 2,792 steps were needed for FClust's convergence, only 122 iterations were



(a) Dataset.

(b) Data clustered into $K = 8$ clusters using K-means.

(c) Agent clusters after applying thresholding.

(d) Data clusters after applying thresholding.

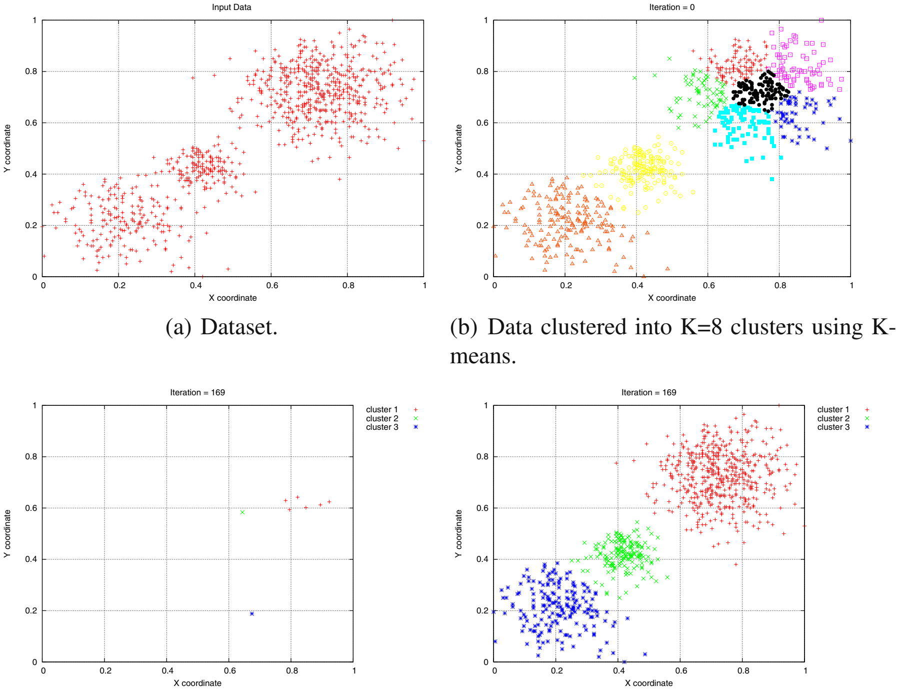
Fig. 10 Stable output of (K-means+FClust) hybrid on a 2 cluster-data set where sim_{th}=0.88 (using Eqn.(2)), d_{th}= 0.4. d_{ideal_th} for FClust=0.04, d_{ideal_th} for forming clusters = 0.08

needed for the Hybrid version. Although the hybrid version speeds up the process considerably, it does not necessarily suffer from the bridging effect observed during the cluster extraction in post-processing.

Figure 11 is a collection of figures showing the results of the (K-means+FClust) Hybrid Algorithm given in Algorithm 7. for Dataset II. When the results are compared with the simple FClust Algorithm in Figures 3 and 4, it can be observed that the hybrid algorithm is faster thanks to fewer iterations and to the modest linear computational cost.

6.3.4 FClust, FClust-Annealing and (K-means+FClust) Hybrid Results for Iris and Pima Datasets

Table 4 compares the FClust, FClust-annealing, and (K-Means+FClust) Hybrid algorithms on the datasets Iris and Pima, which were also used in [30]. The results were averaged over 10 different runs, and the numbers inside the parentheses represent the standard deviations. We have also observed, in the Iris dataset experiments, that the original FClust algorithm tends to get stuck in local optima for a long time, but after enough iterations, it can find a better clustering. For example, in



(a) Dataset.

(b) Data clustered into K=8 clusters using K-means.

(c) Agent clusters after applying thresholding.

(d) Data clusters after applying thresholding.

Fig. 11 Stable output of (K-means+FClust) hybrid on a 3 cluster-data set where sim_{th}=0.73 (using Eqn.(2)), d_{th}= 1.0. d_{ideal_th} for FClust=0.04, d_{ideal_th} for cluster forming =0.08.

Table 4 Compared Results (Averaged over 10 runs)

				FClust			**FClust Annealing**			**FClust Hybrid**		
Dataset	**Number of Items**	**Number of Attributes**	**Number of Classes**	**Average Clusters Found**	**Avg. Error**	**Avg. Iter. No**	**Average Clusters Found**	**Avg. Error**	**Avg. Iter. No**	**Average Clusters Found**	**Avg. Error**	**Avg. Iter. No**
Iris	150	4	3	3.1 (0.32)	0.18 (0.04)	1811 (400.1)	2.9 (0.32)	0.16 (0.03)	11.6 (4.33)	4.5 (0.85)	0.22 (0.06)	41.5 (22.45)
Pima	768	8	2	2 (0)	0.47 (0.01)	3995.1 (984.6)	2 (0)	0.475 (0.02)	19.9 (4.95)	3.8 (1.5)	0.45 (0.03)	41.7 (18.58)

one run, FClust produced 4 clusters around 1500 iterations, with a cluster error rate of approximately 0.25, computed via Algorithm 11, while around 2500 iterations, it formed 3 clusters. However, the error rate was still high (0.27). Later, the number of clusters dropped to 2 and around 7000 iterations, it found 3 clusters with an error rate of 0.08. More iterations resulted in high error rates (around 0.22) again with 2 clusters. So the system will cycle between 2, 3 and 4 clusters. These are different but somehow stable clustering options. However, it was observed that, results with more than 5 clusters with the given parameters, are produced due to an insufficient number of iterations. Again as we have noted in Section 5.4, since every pairwise ideal distance is one constraint/objective to be satisfied, there are $n \times (n-1)/2$ objectives. Theoretically this can lead to up to $n \times (n-1)/2$ Pareto solutions on the Pareto front. The actual number in practice is much less however, since several of these constraints can be satisfied simultaneously.

The FClust-annealing version decreases the number of iterations drastically. It starts forming cluster centers from the first iterations, and even though the centers are created in less than 10 or 20 steps, they still seem to be reasonably good. For the first few iterations, the cluster centers changed rapidly since the neighborhoods were wide. Yet clusters were still formed after post-processing. Finally, the clustering scheme which gave the minimum error was reported. Annealing converges to a meaningful cluster formation faster than FClust. Moreover, in just a few iterations, it can already present different possible clustering options. This process and the changes in the formed cluster numbers and their errors with the iterations, for 10 different runs on the Iris dataset, are shown in Table 5. For each run, the first row is the number of clusters generated at the corresponding iteration number, which is given as the column label. Similarly, the second row shows the error computed via Algorithm 11.

In FClust-annealing and the (K-means+FClust) Hybrid, since the neighborhood is wider, clusters are formed faster, and there will be fewer agents on the visualization panel between agent flocks. Later (in the case of annealing), when the neighborhoods become narrow, the chance of flocks meeting and affecting each other decreases, which causes a decrease in the exploration for better clustering options. Therefore FClust-annealing and the (K-means+FClust) Hybrid are more prone to getting stuck in local optima. Some random moves could be added to the algorithm to increase the opportunity for exploration. One of the problems observed with the (K-means+FClust) Hybrid on the Iris data set was that the algorithm may separate the members of the first cluster into two groups. During the clustering using the Hybrid algorithm, K was set to 8 in K-means, d_{ideal_th} was 0.04, and during post processing, it was 0.08.

To stop the Hybrid algorithm, the *ideal distance error*, for each pair of agents is computed, and when its difference compared to the previous iteration fell below 0.009, the algorithm was stopped. Figure 12 shows the ideal distance error versus the iteration number for the Iris dataset using the (K-means+FClust) Hybrid algorithm. The irregularities observed as sudden increases in the error correspond to the time when clusters reached the border of the visualization panel and continued moving, thus wrapping around toward the opposite side of the panel. That said,

Table 5 Number of clusters extracted and corresponding error at different iteration steps for 10 different runs of clustering the Iris data using FClust-annealing where d_{th}=0.4, d_{ideal_th}=0.04

Iteration No	1	10	50	100	200	300	400	500	1000	1500	2000	2500
Run 1	1 0.67	3 0.16	6 0.26	6 0.26	5 0.24	6 0.24	5 0.18	4 0.16	4 0.15	4 0.15	3 0.13	4 0.15
Run 2	0	3 0.15	2 0.23	2 0.23	2 0.23	2 0.23	2 0.23	2 0.22	2 0.23	2 0.24	3 0.22	2 0.22
Run 3	0	1 0.67	5 0.23	5 0.21	7 0.24	7 0.24	5 0.21	4 0.19	4 0.19	4 0.19	2 0.22	3 0.18
Run 4	0	2 0.24	2 0.23	2 0.22	2 0.22	2 0.23	2 0.22	3 0.17	3 0.17	4 0.18	4 0.19	3 0.09
Run 5	0	2 0.32	4 0.23	4 0.27	4 0.27	4 0.28	4 0.28	4 0.28	2 0.23	3 0.18	3 0.18	4 0.11
Run 6	1 0.67	0	4 0.28	4 0.2	3 0.17	3 0.18	4 0.27	3 0.22	5 0.21	5 0.22	4 0.18	4 0.19
Run 7	0	1 0.67	3 0.19	5 0.16	4 0.14	4 0.17	3 0.15	3 0.16	3 0.15	4 0.1	4 0.13	4 0.11
Run 8	0	0	5 0.26	6 0.28	4 0.22	5 0.2	4 0.22	4 0.22	4 0.26	2 0.22	2 0.22	2 0.22
Run 9	1 0.67	0	5 0.19	3 0.13	3 0.15	3 0.13	3 0.14	3 0.14	2 0.22	2 0.22	2 0.22	2 0.22
Run 10	0	1 0.67	4 0.27	4 0.22	4 0.24	6 0.22	4 0.13	3 0.16	3 0.17	2 0.22	2 0.23	3 0.1
Avg. No of Clusters Average Error	0.3 0.67	1.3 0.411	4 0.237	4.1 0.218	3.8 0.212	4.2 0.212	3.6 0.203	3.3 0.192	3.2 0.198	3.2 0.192	2.9 0.192	3.1 0.159

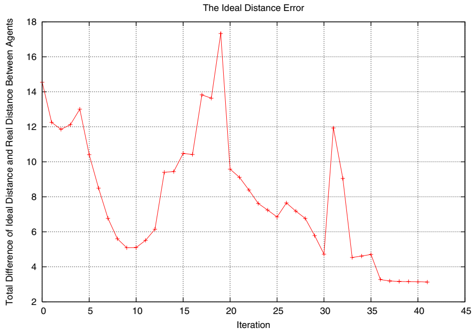


Fig. 12 Evolution of the ideal distance error with iterations for clustering the Iris dataset using (K-means+FClust) Hybrid algorithm. Big changes correspond to major changes due to agents wrapping around the agent space boundaries

the (K-means+FClust) Hybrid algorithm produced reasonable clustering results in fewer iterations on the Iris dataset.

In the experiments with the Pima data, FClust formed 2 or 3 clusters, for d_{th}=0.04 with an error rate around 0.50. Manual stopping terminated earlier than automated

Table 6 Average result of 10 different runs of clustering the Pima data set using FClust-annealing, where d_{th} started from 1 and decreased to 0.04

Found Clusters	Error	Iteration No
2	0.47	1651.1
(0)	(0.01)	(587.3)

stopping because, a Human typically follows the bigger flocks of agents, thus the agents which are spread around the visualization panel do not affect the human observer as much as they may affect the automated stopping mechanism. In the FClust-Annealing algorithm, 2 clusters of the Pima dataset were formed around 20 to 30 iterations, with a cluster error rate between 0.45 and 0.50. For the Pima data set, with smaller cluster size threshold values, clusters would be observed in fewer iterations. However, we kept this threshold constant as $n/20$ to be able to compare the proposed algorithms with the original FClust algorithm. Table 6 shows that an average of 1651 iterations for FClust-annealing were sufficient to get a state of the visualization panel which would require an average of 3995 iterations of the original FClust algorithm. Similarly, K was 8 and d_{ideal_th} was 0.04 for the the Hybrid algorithm. During post processing, d_{ideal_th} was 0.08. We also observed that, for both the Iris and Pima data sets, minimum cluster errors were observed for the FClust-annealing experiments (0.11 for Iris and 0.43 for Pima) compared to all other algorithms.

7 Conclusions and Future Work

Since the early 90s, swarm intelligence (SI) has been a source of inspiration for clustering problems. SI has been used in many applications ranging from image clustering to social clustering, and from document clustering to Web session clustering. Particle swarm clustering and ant-clustering methods are two of the most common SI clustering techniques. But more recently, clustering using flocks of agents proved to be promising as well. Since in this approach, each agent represents a data item, and the distance between agents on the visualization panel depends on the similarity between the data records represented by these agents, the flocks-of-agents approach offers a solution not only for clustering but also for data visualization. Although the initial experimental results with FClust were acceptable, some limitations were discussed in this chapter.

To overcome these limitations, we have proposed some improvements including FClust-annealing and the (K-means+FClust) Hybrid algorithms. Annealing decreased the number of iterations to convergence and improved the quality of the clusters. The effect of different cooling functions should be studied further. In addition to these improvements, the proposed (K-means+FClust) hybrid algorithm reduces the quadratic complexity of FClust to linear complexity and performs similarly to FClust with fewer iterations. In the future, we will consider hybridizing

FClust with the Spherical K-Means algorithm [8] for clustering high-dimensional data such as web usage data and text documents.

Our experiments have illustrated how the cluster formation algorithm was susceptible to the bridging effect for overlapping clusters, and seems to be very sensitive to threshold parameter tuning. Therefore, future studies are needed to devise better automated stopping criteria and algorithms to form better clusters given a state of the visualization panel.

Acknowledgements. This work is partially supported by National Science Foundation CAREER Award IIS-0133948.

References

1. Abraham, A., Das, S., Roy, S.: Swarm Intelligence Algorithms for Data Clustering. In: Soft Computing for Knowledge Discovery and Data Mining, pp. 279–313. Springer, US (2008)
2. Abraham, A., Das, S., Roy, S.: Swarm Intelligence Algorithms for Data Clustering. In: Soft Computing for Knowledge Discovery and Data Mining, pp. 279–313. Springer, US (2008)
3. Azzag, H., Monmarche, N., Slimane, M., Venturini, G.: Anttree: a new model for clustering with artificial ants. In: The 2003 Congress on Evolutionary Computation CEC 2003, vol. 4, pp. 2642–2647 (2003)
4. Couzin, I.D., Krause, J.E.N.S., James, R., Ruxton, G.D., Franks, N.R.: Collective memory and spatial sorting in animal groups. Journal of Theoretical Biology 218(1), 1–11 (2002)
5. Cui, X., Potok, T.E.: Document clustering analysis based on hybrid pso+kmeans algorithm. Journal of Computer Sciences (Special Issue), 27–33 (2005)
6. Cui, X., Potok, T.E.: A distributed agent implementation of multiple species flocking model for document partitioning clustering. In: Klusch, M., Rovatsos, M., Payne, T.R. (eds.) CIA 2006. LNCS, vol. 4149, pp. 124–137. Springer, Heidelberg (2006)
7. Cui, X., Potok, T.E., Palathingal, P.: Document clustering using particle swarm optimization. In: IEEE Swarm Intelligence Symposium (2005)
8. Dhillon, I.S., Modha, D.S.: Concept decompositions for large sparse text data using clustering. Machine Learning 42(1-2), 143–175 (2001)
9. Handl, J., Knowles, J., Dorigo, M.: On the performance of ant-based clustering. In: Proceedings of the Third International Conference on Hybrid Intelligent Systems (2003)
10. Handl, J., Knowles, J., Dorigo, M.: Strategies for the increased robustness of ant-based clustering. In: Di Marzo Serugendo, G., Karageorgos, A., Rana, O.F., Zambonelli, F. (eds.) ESOA 2003. LNCS, vol. 2977, pp. 90–104. Springer, Heidelberg (2004)
11. Handl, J., Meyer, B.: Ant-based and swarm-based clustering. Swarm Intelligence 1(2), 95–113 (2007)
12. Heppner, F., Grenander, U.: A stochastic nonlinear model for coordinated bird flocks. In: Krasner, S. (ed.) The Ubiquity of Chaos, pp. 233–238. AAAS, Washington (1990)
13. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs (1988)
14. Jain, A.K., Murthy, M., Flynn, P.: Data clustering: A review. ACM Computing Reviews (1999)

15. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
16. Kennedy, J., Eberhart, R., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers Inc, San Francisco (2001)
17. Labroche, N., Monmarche, N., Venturini, G.: A new clustering algorithm based on the chemical recognition system of ants. In: Proceedings of the 15th European Conference on Artificial Intelligence (2002)
18. Labroche, N., Monmarche, N., Venturini, G.: Antclust: Ant clustering and web usage mining. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 25–36. Springer, Heidelberg (2003)
19. Labroche, N., Monmarche, N., Venturini, G.: Web sessions clustering with artificial ants colonies. In: WWW 2003, The Twelfth International World Wide Web Conference, Budapest, Hungary (2003)
20. Lumer, E.D., Faieta, B.: Diversity and adaptation in populations of clustering ants. In: Proceedings of the third international conference on Simulation of adaptive behavior: from animals to animats 3, pp. 501–508. MIT Press, Cambridge (1994)
21. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Cam, L.M.L., Neyman, J. (eds.) Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press (1967)
22. van der Merwe, D.W., Engelbrecht, A.P.: Data clustering using particle swarm optimization. In: Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003), vol. 1, pp. 215–220 (2003)
23. Millonas, M.M.: Swarms, phase transition, and collective intelligence. In: Langton, C.G. (ed.) Artificial life III. Addison Wesley, Reading (1994)
24. Nasraoui, O., Krishnapuram, R., Frigui, H.: Extracting web user profiles using relational competitive fuzzy clustering. International Journal on Artificial Intelligence Tools 9(4), 509–526 (2000)
25. Nasraoui, O., Krishnapuram, R., Joshi, A.: Mining web access logs using a relational clustering algorithm based on a robust estimator. In: Proc. of the Eighth International World Wide Web Conference, Toronto, pp. 40–41 (1999)
26. Nasraoui, O., Krishnapuram, R., Joshi, A.: Relational clustering based on a new robust estimator with application to web mining. In: Proceedings of the North American Fuzzy Information Society, New York City, pp. 705–709 (1999)
27. Omran, M., Engelbrecht, A.P., Salman, A.: Particle swarm optimization method for image clustering. International Journal of Pattern Recognition and Artificial Intelligence 19(3), 297–322 (2005)
28. Omran, M., Salman, A., Engelbrecht, A.P.: Image classification using particle swarm optimization. In: Conference on Simulated Evolution and Learning, vol. 1, pp. 370–374 (2002)
29. Picarougne, F., Azzag, H., Venturini, G., Guinot, C.: On data clustering with a flock of artificial agents. In: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2004 (2004)
30. Picarougne, F., Azzag, H., Venturini, G., Guinot, C.: A new approach of data clustering using a flock of agents. Evolutionary Computation 15(3), 345–367 (2007)
31. Proctor, G., Winter, C.: Information flocking: Data visualisation in virtual worlds using emergent behaviours. In: Heudin, J.-C. (ed.) VW 1998. LNCS, vol. 1434, pp. 168–176. Springer, Heidelberg (1998)
32. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. Computer Graphics 21(4), 25–34 (1987)

33. Saka, E., Nasraoui, O.: Simultaneous clustering and visualization of web usage data using swarm-based intelligence. In: Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2008 (2008)
34. Vizine, A.L., de Castro, L.N., Hruschka, E.R., Gudwin, R.R.: Towards improving clustering ants: an adaptive ant clustering algorithm. Informatica 29, 143–154 (2005)
35. Weiss, G. (ed.): Multiagent Systems: A Modern Approach To Distributed Artificial Intelligence. The MIT Press, Cambridge (2000)
36. White, T., Pagurek, B.: Towards multi-swarm problem solving in networks. In: Demazeau, Y. (ed.) Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS 1998). IEEE Press, Paris (1998)

Combining Statistics and Case-Based Reasoning for Medical Research

Rainer Schmidt and Olga Vorobieva

Abstract. In medicine many exceptions occur. In medical practice and in knowledge-based systems too, it is necessary to consider them and to deal with them appropriately. In medical studies and in research, exceptions should be explained. We present a system, called ISOR, that helps to explain cases that do not fit to a theoretical hypothesis. Starting points are situations where neither a well-developed theory nor reliable knowledge nor, at the beginning, a case base is available. So, instead of theoretical knowledge and intelligent experience, just some theoretical hypothesis and a set of measurements are given. In this chapter, we focus on the application of the ISOR system to the hypothesis that a specific exercise program improves the physical condition of dialysis patients. Additionally, for this application a method to restore missing data is presented.

1 Introduction

Case-based Reasoning (CBR) uses previous experience represented as cases to understand and solve new problems. A case-based reasoner remembers former cases similar to the current problem and attempts to modify solutions of former cases to fit the current problem.

The fundamental ideas of CBR originated in the late eighties (e.g,. [24]). In the early nineties CBR emerged as a method that was firstly described by Kolodner [14]. Later on, Aamodt and Plaza presented a more formal characterisation of the CBR method. Figure 1 shows the Case-based Reasoning cycle developed by Aamodt and Plaza [1], which consists of four steps: retrieving former similar cases, adapting their solutions to the current problem, revising a proposed solution, and retaining

Rainer Shmidt
Institute for Medical Informatics and Biometry, University of Rostock, Germany
e-mail: rainer.schmidt@medizin.uni-rostock.de

Olga Vorobieva
Institute for Medical Informatics and Biometry (as above) and Sechenow
Institute of Evolutionary Physiolonary and Biochemistry, St.Petersburg, Russia

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 673–696.
springerlink.com

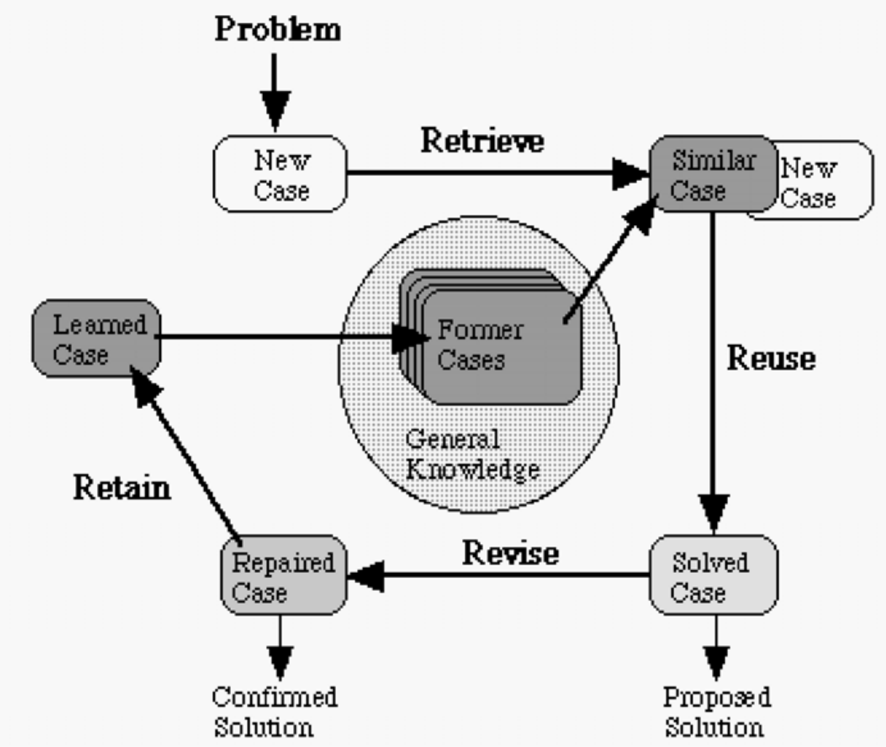


Fig. 1 The Case-based Reasoning cycle developed by Aamodt

new learned cases. However, there are two main subtasks in Case-based Reasoning [14, 1]: The retrieval (the search for a similar case), and the adaptation (the modification of solutions of retrieved cases). For retrieval, many similarity measures and sophisticated retrieval algorithms have been developed within the CBR community. The most common ones are indexing methods [14] like tree-hash retrieval [28], which are useful for nominal parameter values, retrieval nets [15], which are useful for ordered nominal values, and nearest neighbour search [5], which is useful for metric parameter values.

The second task, the adaptation, is a modification of solutions of former similar cases to fit for a current one. If there are no important differences between a current and a similar case, a simple solution transfer is sufficient. Sometimes only few substitutions are required, but at other times the adaptation is a very complicated process. So far, for adaptation only very domain independent methods like compositional adaptation [29] currently exist.

1.1 Case-Based Reasoning in Medicine

Especially in medicine, the knowledge of experts does not only consist of rules, but of a mixture of textbook knowledge and experience. The latter consists of cases, typical and exceptional ones, and the reasoning of physicians takes them into account [10]. In medical knowledge based systems there are two types of knowledge, objective knowledge, which can be found in textbooks, and subjective knowledge, which is limited in space and time and changes frequently.

The problem of updating the changeable subjective knowledge can partly be solved by incrementally incorporating new up-to-date cases [10]. Both kinds of knowledge can be clearly separated: Objective textbook knowledge can be represented in forms of rules or functions, while subjective knowledge is contained in cases.

So, the arguments for case-oriented methods are as follows:

1. Reasoning with cases corresponds with the decision making process of physicians.
2. Incorporating new cases means automatically updating parts of the changeable knowledge.
3. Objective and subjective knowledge can be clearly separated.
4. As cases are routinely stored, integration into clinic communication systems is easy.

Since differences between two cases are sometimes very complex, especially in medical domains, many case-based systems are so called retrieval-only systems. They only perform the retrieval task, visualise current and similar cases, and sometimes additionally point out the important differences between them.

However, a string of medical CBR systems have already been developed, for overviews see [11, 25, 20].

1.2 The ISOR Approach

In our previous work on knowledge-based systems [26], it is demonstrated how a dialogue-oriented Case-Based Reasoning system can help in situations where a theoretically approved medical therapy does not produce the desired and normally expected results. In medical studies and in research, exceptions need to be explained. We have developed ISOR, a conversational case-based system that helps doctors to explain exceptional cases. Conversational CBR systems have begun to become popular in recent years [2]. However, so far no common conversational methodology exists, only the common idea to incorporate the user into a conversational solution search.

ISOR deals with situations where neither a well-developed theory nor reliable knowledge nor a proper case base is available. So, instead of theoretical knowledge and intelligent experience, just a theoretical hypothesis and a set of measurements are given. In such situations the usual question is, "how do measured data fit to a theoretical hypothesis?" To statistically confirm a hypothesis it is necessary that the majority of cases fit the hypothesis. Mathematical statistics determines the exact quantity of necessary confirmation [13]. However, when a few cases do not satisfy the hypothesis, these cases need to be examined to find out why they do not fit.

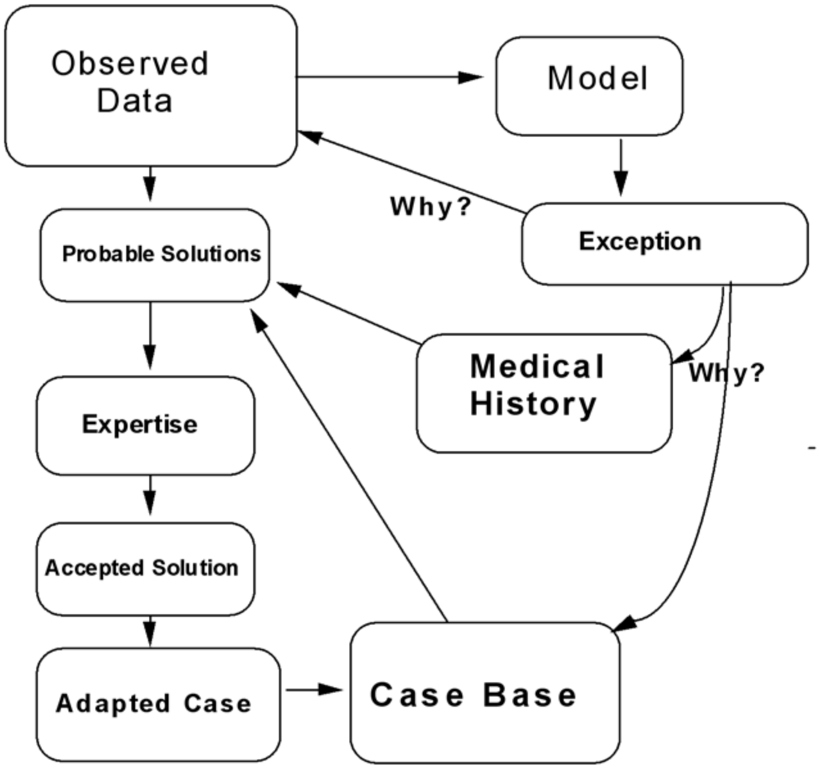


Fig. 2 ISOR's general program flow

ISOR offers a dialogue to guide the search for possible reasons in all components of the data system. The exceptional cases belong to the case base. This approach is justified by a certain mistrust of statistical models by doctors, because modelling results are usually non-specific and "average oriented" [12], which reflects a lack of attention to individual "imperceptible" features of specific patients.

The usual Case-Based Reasoning assumption is that a case base with complete solutions is available [14, 1, 21]. Our approach starts with a situation where such a case base is not available but has to be set up incrementally. The general program flow is shown in Figure 2. The main steps are:

1. Construct a model,
2. Point out the exceptions,
3. Find causes why the exceptional cases do not fit the model, and
4. Set up a case base.

So, Case-Based Reasoning is combined with a model, in this specific situation with a statistical one. The idea to combine CBR with other methods is not new. Care-Partner, for example, resorts to a multi-modal reasoning framework for the co-operation of CBR and rule-based reasoning (RBR) [4]. Montani [19] rather uses

CBR to provide evidence for a hybrid system in the domain of diabetes. Another way of combining hybrid rule bases with CBR is discussed by Prentzas and Hatzilgeroudis [22]. The combination of CBR and model-based reasoning is discussed in [27]. However, statistical methods are used within CBR mainly for retrieval and retention [6, 23]. Arshadi and Jurisica [3] propose a method that combines CBR with statistical methods such as clustering and logistic regression.

The first application of ISOR is on hemodialysis and fitness. Unfortunately, the data set contains many missing data items, which makes the process of finding explanations for exceptional cases difficult. So, we decided to attempt to first solve the missing data problem. This is done by partly applying CBR again.

Hemodialysis means stress for a patient's organism and has significant adverse effects. Fitness is the most available and a relative cheap way of support. It is meant to improve a physiological condition of a patient and to compensate negative dialysis effects. One of the intended goals of this research is to convince patients of the positive effects of fitness and to encourage them to actively participate in the fitness program. This is important because dialysis patients usually feel sick, they are physically weak, and they do not want any additional physical load [7].

At the University clinic in St. Petersburg, a specially developed complex of physiotherapy exercises including simulators, walking, swimming and so on, is offered to all dialysis patients. However, only some of them *actively* participate, whereas some others participate but are not really active. The purpose of this fitness offer is to improve the physical conditions of the patients and to increase the quality of their lives. The hypothesis is that actively participating in the fitness program improves the physical condition of dialysis patients.

2 Incremental Development of an Explanation Model for Exceptional Dialysis Patients

For each patient a set of physiological parameters is measured. These parameters contain information about burned calories, maximal power achieved by the patient, oxygen uptake, oxygen pulse (volume of oxygen consumption per heart beat), lung ventilation and others. There are also biochemical parameters like haemoglobin and other laboratory measurements. More than 100 parameters were planned for every patient. But not all of them were actually measured.

Parameters are supposed to be measured four times during the first year of participating in the fitness program. There is an initial measurement followed by one after three months, then after six months and finally after a year. Unfortunately, since some measurements were not taken, many data are missing. Therefore the patient records often contain different sets of measured parameters.

It is necessary to note that parameter values of dialysis patients essentially differ from those of non-dialysis patients, especially of healthy people, because dialysis interferes with the natural, physiological processes in an organism. In fact, for dialysis patients, all physiological processes behave abnormally. Therefore, the correlation between parameters differs too.

For statistics, this means difficulties in applying statistical methods based on correlation and it limits the usage of a knowledge base developed for normal people. Non-homogeneity of observed data, many missing data, many parameters for a relatively small sample size, all this makes the data set practically impossible for usual statistical analysis.

Since the data set is incomplete, additional or substitutional information has to be found from other available data sources. These are databases – the already existent individual base and the sequentially created case base – and the medical expert as a special source of information.

2.1 Setting up a Model

A medical problem needs to be solved based on given data. In this example it is: *”Does special fitness improve the physiological condition of dialysis patients?”* More formally, physical conditions of active and non-active patients need to be compared. Patients are divided into two groups, depending on their activity, active and non-active patients.

According to the assumption, active patients should feel better after some months of the fitness program, whereas non-active ones should feel rather worse. The meaning of “feeling better” and “feeling worse” has to be defined in this context. Therefore, a medical expert selects appropriate factors from ISOR’s menu, which contains the parameter names from the observed database. The expert selects the following main factors:

- F1: O2PT - Oxygen pulse by training
- F2: MUO2T - Maximal Uptake of Oxygen by training
- F3: WorkJ – performed Work (Joules) during control training

Subsequently the “research time period” has to be determined. Initially, this period was planned to be twelve months, but after a while the patients tend to give up the fitness program. This means, the longer the time period, the more data are missing. Therefore, a compromise between time period and sample size had to be made; a period of six months was chosen.

The next question is whether the model should be quantitative or qualitative? The observed data are mostly quantitative measurements. The selected factors are also quantitative in nature. On the other hand, the goal of this research is to find out whether physical training improves or worsens the physical condition of dialysis patients.

One patient does not have to be compared with another patient. Instead, each patient has to be compared with his/her own situation some months ago, namely just before the start of the fitness program. The success should not be measured in absolute values, because the health statuses of patients are very different. Thus, even a modest improvement for one patient may be as important as the great improvement of another. Therefore, we simply classify the development into two categories: “better” and “worse”. Since the usual tendency for dialysis patients is to

Table 1 Results of Fisher's exact test, performed via an interactive web-program: http://www.langsrud.com/fisher.htm. The cases in bold have to be explained

Improvement mode	Patient's physical condition	Active	Non-active	Fisher Exact p
Strong	Better	28	1	$<$ 0.0001
	Worse	**22**	21	
Medium	Better	40	10	$<$ 0.005
	Worse	**10**	12	
Weak	Better	47	16	$<$ 0.02
	Worse	**3**	6	

worsen over time, those few patients where no changes could be observed are added to the category "better".

The three main factors are supposed to describe the changes of the physical conditions of the patients. The changes are assessed depending on the number of improved factors:

- Weak version of the model: at least one factor has improved
- Medium version of the model: at least two factors have improved
- Strong version of the model: all three factors have improved

The final step is to define the type of model. Popular statistical programs offer a large variety of statistical models. Some of them deal with categorical data. The easiest model is a 2x2 frequency table. The "better/ worse" concept fits this simple model very well. So the 2x2 frequency table is accepted. The results are presented in Table 1. According to the assumption after six months of active fitness the conditions of the patients should be better.

Statistical analysis shows a significant dependence between the patients activity and improvement of their physical condition. Unfortunately, the most popular Pearson Chi-square test is not applicable here because of the small values "2" and "3" in Table 1. But Fisher's exact test [13] can be used. In the three versions shown in Table 1 a very strong significance can be observed. The smaller the value of p is, the more significant the dependency.

Exceptions. The performed Fisher test confirms the hypothesis that patients doing active fitness achieve better physical conditions than non-active ones. However, there are exceptions, namely active patients whose health conditions did not improve.

Exceptions need to be explained. Explained exceptions build the case base. According to Table 1, the stronger the model, the more exceptions can be observed and have to be explained. Every exception is associated with at least two problems. The first one is "Why did the patient's condition get worse?" Of course, "worse" is meant in terms of the chosen model. Since there may be some factors that are not included in the model but have changed positively, the second problem is "What has improved in the patient's condition?" To solve this problem significant factors where the values improved have to be searched.

In the following section the set-up of a case base on the strongest model version is explained.

2.2 Setting up a Case Base

We intend to solve both problems (mentioned above) by means of CBR. So we begin to set up the case base up sequentially. That means, as soon as an exception is explained, it is incorporated into the case base and can be used to help explain further exceptional cases. Alphabetical order for the exceptional cases was chosen.

The retrieval of already explained cases is performed by keywords. The main keywords are "problem code", "diagnosis", and "therapy". In the situation of explaining exceptions for dialysis patients, the instantiations of these keywords are "adverse effects of dialysis" (diagnosis), "fitness" (therapy), and two specific problem codes. Besides the main keywords, additional problem specific ones are used. Here the additional keyword is the number of worsened factors. Further keywords are optional. They are just used when the case base becomes bigger and retrieval is no longer simple.

However, ISOR not only uses the case base as a knowledge source but further sources are involved, namely the patient's individual base (his medical history) and observed data (partly gained by dialogue with medical experts). Since in the domain of kidney disease and dialysis the medical knowledge is very detailed and much investigated but still incomplete, it is unreasonable to attempt to create an adequate knowledge base. Therefore, a medical expert, observed data, and just a few rules serve as medical knowledge sources.

2.2.1 Expert Knowledge and Artificial Cases

An expert's knowledge can be used in many different ways. Firstly, it is used to acquire rules, and secondly, it can be used to select appropriate items from the list of retrieved solutions, to propose new solutions, and last but not least, to create artificial cases.

Initially, artificial cases are created by an expert, afterwards they can be used in the same way as real cases. They are created in the following situation. An expert points out a factor F as a possible solution for a query patient. Since many data are missing, it may happen that just for the query patient values of factor F are missing. In such a situation the doctor's knowledge can not be applied, but it is sensible to save it anyway. In essence, there are two different ways to do this. The first one is to generate a correspondent rule and to insert it into ISOR's algorithms. Unfortunately, this is very complicated, especially to find an appropriate way for inserting such a rule. The alternative is to create an artificial case. Instead of the name of a patient an artificial case number is generated. The other attributes are either inherited from the query case or declared as missing. The retrieval attributes are inherited. This can be done by a short dialogue (Figure 4) and ISOR's algorithms remain intact. Artificial cases can be treated in the same way as real cases: they can be revised, deleted, generalised and so on.

2.2.2 Solving the Problem "Why Did Some Patients Conditions Became Worse?"

A set of solutions of different origin and different nature is obtained. There are three categories of solutions: additional factor, model failure, and wrong data.

Additional factor. The most important and most frequent solution is the influence of an additional factor. However, three main factors are obviously not enough to describe all medical cases. Unfortunately, for different patients different additional factors are important. When ISOR has discovered an additional factor as explanation for an exceptional case, the factor has to be confirmed by the medical expert before it can be accepted as a solution. One of these factors is Parathyroid Hormone (PTH). An increased PTH level can sometimes explain a worsened condition of a patient [7]. PTH is a significant factor, but unfortunately it was measured for only some patients.

Another additional factor as a solution is blood phosphorus level. The principle of artificial cases was used to introduce the factor phosphorus as a new solution. One patient's record contained many missing data. The retrieved solution meant high PTH, but PTH data in the current patient's record was missing too. The expert proposed an increased phosphorus level as a possible solution. Since data about phosphorus data was also missing, an artificial case was created that inherited all retrieval attributes of the query case, whereas the other attributes were recorded as missing. According to the expert, high phosphorus can provide an explanation. Therefore it is accepted as an artificial solution or a solution of an artificial case.

Some exceptions can be explained by indirect indications, which can be considered as another sort of additional factor. One of them is a very long period of dialysis (more than 60 months) before a patient began with the fitness program.

Model failure. We regard two types of model failures. One of them is deliberately neglected data. As a compromise we only considered data collected in the chosen six months period, whereas further data of a patient might be important. In fact, three of the patients did not show an improvement in the considered six months, but did so in the following six months. So, they were wrongly classified and should really belong to the "better" category. The second type of model failure is based on the fact that the two-category model was not precise enough. Some exceptions could be explained by a tiny and not really significant change in one of the main factors.

Wrong data are usually due to a technical mistake or to data not really proved. One patient, for example, was reported as actively participating in the fitness program but really was not.

2.2.3 Solving the Problem "What in the Patient's Condition Improved?"

There are at least two criteria to select factors for the model. First, a factor has to be significant, and second there must be enough patients for which this factor was measured at least for six months. So, some principally important factors were initially not taken into account because of missing data. The list of solutions includes these factors (Figure 4): haemoglobin and maximal power (watt) achieved during control training. Oxygen pulse and oxygen uptake were measured in two different situations, namely during the training and before training in a state of rest. Therefore we have two pairs of factors: oxygen pulse in state of rest (O2PR) and during training (O2PT); maximal oxygen uptake in state of rest (MUO2R) and during training (MUO2T). Measurements made in a state of rest are more indicative and significant than those made during training. Unfortunately, most measurements were made during training. Only for some patients did corresponding measurements in a state of rest exist. Therefore O2PT and MUO2T were accepted as main factors and were incorporated into the model. On the other hand, O2PR and MUO2R are solutions for the current problem "What in the patient's condition improved?"

In the case base every patient is represented by a set of cases, and every case represents a specific problem. This means that a patient is described from different points of view, and accordingly different keywords are used for retrieval.

2.3 Another Problem

Based on the same data set, with this method and this dialogue menu (Figure 4) many research questions can be analysed. Above we described just one of them. Another interesting research question is "Does it make sense to start the fitness program during the first year of dialysis?" The question arises, because the conditions of the patients are considered to be unstable during their first year of dialysis. So, the question is expressed in this way "When should patients begin with the fitness program, earlier or later?" The term "earlier" is defined as "during the first year of dialysis". The term "later" means after at least one year of dialysis. To answer this question two groups of active patients are considered, namely those who began their training within the first year of dialysis and those who began it later (Table 2).

According to Fisher's exact test a dependence can be observed, with $p < 0.05$. However, the result is not as it was initially expected. Since patients are considered as unstable during their first year of dialysis, the assumption was that an earlier beginning might worsen conditions of the patients. But the test revealed that the conditions of active patients who began with their fitness program within the first year of dialysis improved even more than the conditions of patients starting later.

Table 2 Changed conditions for active patients

	Earlier	Later
Better	18	10
Worse	6	16

However, there are six exceptional cases, namely those active patients starting early and their conditions worsened. These exceptions belong to the case base, the explanations of them are high PTH or high phosphorus level.

2.4 *Example*

The following example demonstrates how ISOR attempts to find explanations for exceptional cases. Because of data protection no real patient can be used. It is an artificial case but nevertheless it is a typical situation.

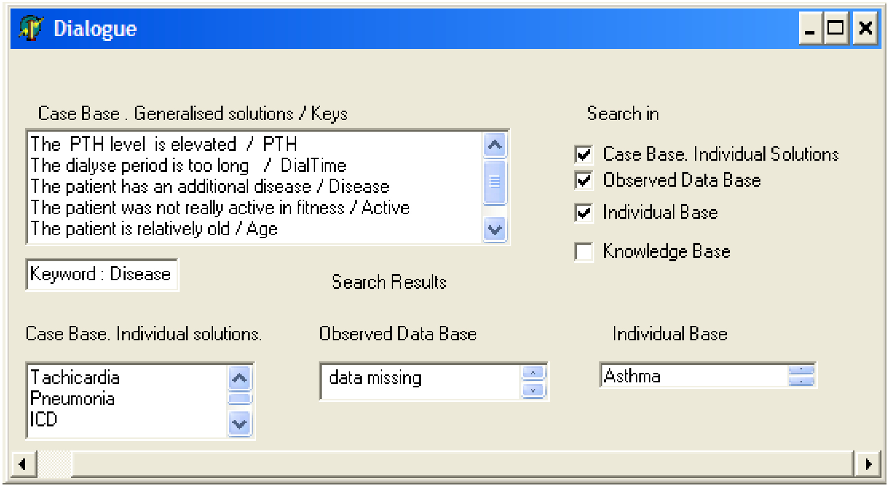
Query patient: a 34-year old woman started with fitness after five months of dialysis. Two factors worsened, namely Oxygen pulse and Oxygen uptake, and consequently the condition of the patient was assessed as worsened too.

Problem: Why the patient's condition deteriorated after six months of physical training?

Retrieval: The number of worsened factors is used as an additional keyword in order to retrieve all cases with at least two worsened factors.

Case base: It does not only contain cases but more importantly a list of general solutions. For each of the general solutions there exists a list that contains specific solutions based on the cases in the case base. The list of general solutions contains these five items (Figure 3):

1. Concentration of Parathyroid Hormone (PTH),
2. Period of dialysis is too long,
3. An additional disease,



Dialogue

Case Base . Generalised solutions / Keys

The PTH level is elevated / PTH
The dialyse period is too long / DialTime
The patient has an additional disease / Disease
The patient was not really active in fitness / Active
The patient is relatively old / Age

Keyword : Disease

Search in

☑ Case Base. Individual Solutions
☑ Observed Data Base
☑ Individual Base
☐ Knowledge Base

Search Results

Case Base. Individual solutions.

Tachicardia
Pneumonia
ICD

Observed Data Base

data missing

Individual Base

Asthma

Fig. 3 Dialog menu to search for general solution

4. A patient was not very active during the fitness program, and
5. A patient is very old.

Individual base. The patient suffers from a chronic disease, namely from asthma.

Adaptation. Since the patient started with a fitness program already after five months of dialysis, the second general solution can be excluded. The first general solution might be possible, though the individual base does not contain any information about PTH. Further laboratory tests showed PTH = 870, which means that PTH is a solution.

Since an additional disease, bronchial asthma, is found in the individual base, this solution is checked. Asthma is not contained as a solution in the case base, but the expert concludes that asthma can be considered as a solution. Concerning the remaining general solutions, the patient is not too old and proclaims that she was active at fitness.

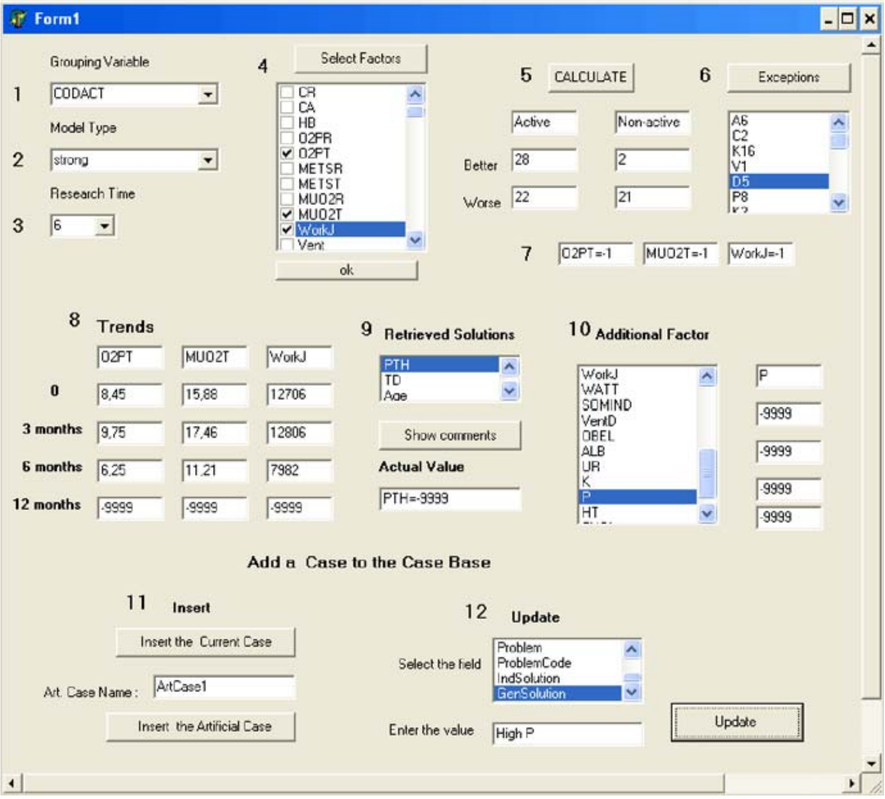
Adapted case. The solution consists of a combination of two factors, namely a high PTH concentration and an additional disease, asthma.

3 Illustration of ISOR's Program Flow

Figure 4 shows the main dialogue of ISOR. At first, the user sets up a model (steps 1 to 4), subsequently he/she gets the result and an analysis of the model (steps 5 to 8), and then he/she attempts to find explanations for the exceptional cases (steps 9 and 10). Finally, the case base is updated (steps 11 and 12). On the menu (Figure 4) the steps are numbered, and in the following paragraph they are explained in detail.

At first the user has to set up a model. To do this he/she has to select a grouping variable. In this example CODACT was chosen. It stands for "activity code", and means that active and non-active patients are to be compared. Provided alternatives are the sex and the beginning of the fitness program (within the first year of dialysis or later). In another menu the user can define further alternatives. Furthermore, the user has to select a model type (alternatives are "strong", "medium", and "weak"), the length of the time period that should be considered (3, 6 or 12 months), and the main factors have to be selected. The list contains the factors from the observed database. In the example, three factors are chosen: O2PT (oxygen pulse by training), MUO2T (maximal oxygen uptake by training), and WorkJ (work in joules during the test training). In the menu list, the first two factors have alternatives: "R" instead of "T", where "R" stands for state of rest and "T" for state of training.

When the user has selected these items, ISOR calculates the table. "Better" and "worse" are meant in the sense of the chosen model, in the case of our example, the strong model was used. ISOR does not only calculate the table but additionally extracts the exceptional patients from the observed database. In the menu, the list of exceptions shows the code names of the patients. In the example, patient "D5" is selected and all further data belong to this patient.



1 Grouping Variable: CODACT

2 Model Type: strong

3 Research Time: 6

4 Select Factors

☐ CR
☐ CA
☐ HB
☐ O2FR
☑ O2PT
☐ METSR
☐ METST
☐ MUO2R
☑ MUO2T
☑ WorkJ
☐ Vent

5 CALCULATE

	Active	Non-active
Better	28	2
Worse	22	21

6 Exceptions

A6
C2
K16
V1
D5
P8

7 O2PT=-1 MUO2T=-1 WorkJ=-1

8 Trends

	O2PT	MUO2T	WorkJ
0	8,45	15,88	12706
3 months	9,75	17,46	12806
6 months	6,25	11,21	7982
12 months	-9999	-9999	-9999

9 Retrieved Solutions

PTH
TD
Age

Actual Value: PTH=-9999

10 Additional Factor

WorkJ
WATT
SOMIND
VentD
OBEL
ALB
UR
K
P
HT

P
-9999
-9999
-9999
-9999

Add a Case to the Case Base

11 Insert

Art. Case Name : ArtCase1

12 Update

Select the field:
Problem
ProblemCode
IndSolution
GenSolution

Enter the value: High P

Fig. 4 ISOR's main dialogue menu

The goal is to find an explanation for the exceptional case "D5". In point 7 of the menu it is shown that all selected factors worsened (-1), and in point 8 the factor values according to different time intervals are depicted. All data for twelve months are missing (-9999).

The next step means creating an explanation for the selected patient "D5". From the case base ISOR retrieves general solutions. The first retrieved solution in this example, the PTH factor, denotes that the increased parathyroid hormone blood level may explain the failure. Further theoretical information (e.g. normal values) about a selected item can be received by pressing the button "show comments". The PTH value of patient "D5" is missing (-9999). From menu point 10 the expert user can select further probable solutions. In the example, an increased phosphorus level (P) is suggested. Unfortunately, phosphorus data are missing too. However, the idea of an increased phosphorus level as a possible solution should not be lost. So, an artificial case should be generated.

The final step means inserting new cases into the case base. There are two kinds of cases, query cases and artificial cases. Query cases are stored records of real patients from the observed database. Artificial cases inherit the key attributes from the query cases (point 7 in the menu). Other data may be declared as missing. Using the update function, the missing data can be inserted later on. In the example of Figure 4, the generalised solution "High P" is inherited, it may be retrieved as a possible solution (point 9 of the menu) for future cases.

4 Missing Data

Databases with many variables have specific problems. Since it is very usually difficult to overview their content, a priory, a user does not know how complete a data set is. Are there any data missing? How many of them and where are they located?

In the dialysis data set, many data are missing in a random fashion, without any regularity. The main cause is that many measurements were not taken.

It can be assumed that the data set contains groups of interdependent variables but a priory it is not known how many such groups there are, what kind of variables are dependent, and in which way they are dependent. However, we intend to make use of all possible forms of dependency to restore missing data, because the more complete the observed data base is, the easier it should be to find explanations for exceptional cases and, furthermore, the better the explanations should be. Even for setting up the model the expert user should select those parameters as main factors with only few missing data. So, the more data that are restored, the better the choice for setting up the model can be.

A data analysis method is often assessed according to its tolerance to missing data (e.g., in [18]). In principle, there are two main approaches to the missing data problem. The first approach is a statistical restoration of missing data. Usually it is based on non-missing data from other records.

The second approach suggests methods that accept the absence of some data. The methods of this approach can be differently advanced, from simply excluding cases with missing values up to rather sophisticated statistical models [17, 8].

Gediga and Düntsch [9] propose the use of CBR to restore missing data. Since their approach does not require any external information, they call it a "non-invasive imputation method". Missing data are supposed to be replaced by their correspondent values of the most similar retrieved cases. However, the dialysis data set contains rather few patients, which means that the "most similar" case for a query case might not be very similar at all.

So, why don't we just apply statistical methods? Statistical methods require homogeneity of the sample. However, there are no reasons to expect the set of dialysis patients to be a homogenous sample. Since the data consists of many parameters, sometimes missing values can be calculated or estimated from other parameter values. Furthermore, the number of cases in the data set is rather small, whereas in general, statistical methods are more appropriate the larger the number of cases.

4.1 The Data Set

For each patient a set of physiological parameters is measured. These parameters contain information about burned calories, maximal power, oxygen pulse (volume of oxygen consumption per heartbeat), lung ventilation, and many others. Furthermore, there are biochemical parameters like haemoglobin and other laboratory measurements. All these parameters are supposed to be measured four times during the first year of participating in the fitness program. There is an initial measurement followed by a next one after three months, then after six months, and finally after a year. Since some parameters, e.g. the height of a patient, are supposed to remain constant within a year, they were measured just once. The other ones are regarded as factors with four grades, they are denoted as F0 – the initial measurement of factor F, and F3, F6, and F12 – the measurements of factor F after 3, 6, and 12 months.

All performed measurements are stored in the observed database, which contains 150 records (one patient – one record) with 460 variables per record. 12 variables are constants, the other 448 variables represent 112 different parameters.

The factors can not be considered as completely independent from each other, but there are different types of dependency among specific factors. Even a strict mathematical dependency can occur, for example the triple: "time of controlled training, work performed during this time, and average achieved power", can be expressed as Power = Work/Time. Less strict are relations between factors of biochemical nature. An increase of parathyroid hormone, for example, implies an increase of blood phosphorus. Such relations between factors enable us to fill some missing data in the data set.

4.2 Restoration of Missing Data

In ISOR, again CBR is applied, now to restore missing data, the calculated values are filled in the observed database. The whole knowledge is contained in the case base, namely in form of solutions of former cases.

4.2.1 Types of Solutions

There are three types of numerical solutions: exact, estimated, and binary. Some examples and restoration formulas are shown in Table 3. All types of solutions are demonstrated by examples in the next section.

When a missing value can be completely restored, it is called an exact solution. Exact solutions are based on other parameters. A medical expert has defined them as specific relations between parameters, using ISOR. As soon as they have been used once, they are stored in the case base of ISOR and can be retrieved for further cases.

Since estimated solutions are usually based on domain independent interpolation, extrapolation, or regression methods, a medical expert is not involved. An estimated solution is not considered as full reconstruction but just as an estimation.

Table 3 Some examples of solutions and restoration formulas. Abbreviations: BC = Breath consumption, BF = Breath frequency, BV = Breath volume, HAT = Hematocrit, P = Phosphorus, PTH = Parathyroid hormone, PV = plasma volume

Missing parameter	Type of solution	Numeric solution (examples)	Description	Parameters	Time points
PTH	Binary	1	If P(T) >= P(t) then PTH(T) >= PTH(t) Else PTH(T) < PTH(t)	P, PTH	0 and 6
HT	Exact	36,2	HT = 100 * (1–PV/0.065 * Weight)	PV, Weight	6
HT	Estimated	29,1	Y(6) = Y(3)*0.66 + Y(12) * 0.33	HT	3 and 12
WorkJ	Exact	30447,1	WorkJ = MaxPower * Time * 0.5	Time, Max-Power	12
BC	Exact	15,6	BC = BF * BV	BF, BV	12
Oxygen pulse	Estimated	10,29	Linear regression	O2plus	0 and 3 and 12

A binary solution is a partial reconstruction of a missing value. Sometimes ISOR is not able to construct either an exact or an estimated solution, but the expert may draw a conclusion about increasing/decreasing the missing value. So, a binary solution expresses just the assumed trend. "1" means that the missing value should have increased since the last measurement, whereas "0" means that it should have decreased. Binary solutions are used in the qualitative models of ISOR.

4.2.2 Examples

The following three typical examples demonstrate how missing data are restored.

First example: Exact solution

The value of hematocrit (HT) after six months is missing. Hematocrit is the proportion of the blood volume that consists of red blood cells. So, the hematocrit measurements are expressed as percentages.

The retrieved solution (the third line of Table 3) requires two additional parameters, namely plasma volume (PV) and the weight of the patient. For the query patient these values (measured after six months) are weight = 74 kg and PV = 3,367. These values are inserted in the formula and the result is a hematocrit value of 30%.

This restoration is domain dependent, it combines three parameters in such a specific way that it can not be applied to any other parameters. However, the formula can of course be transformed in two other ways and so it can be applied to restore values of PV and the weight of the patient. The formula contains specific medical knowledge that was once given as a case solution by an expert.

Second example: Estimated solution
This is the same situation as in the first example. The value of hematocrit that should have been measured after six months is missing. Unlike the first example, now the PV value that is required to apply the domain dependent formula is also missing. Since no other solution for exact calculation can be retrieved, ISOR attempts to generate an estimated solution. Of course, estimated solutions are not as good as exact ones but are acceptable. ISOR retrieves a domain independent formula (fourth line of Table 3) that states that a missing value after six months should be calculated as the sum of two-thirds of the value measured after three months and one-third of the value measured after twelve months. This general calculation can be used for many parameters.

Third example: Binary solution
The value of parathyroid hormone (PTH) after six months is missing and need to be restored. The retrieved solution involves the initial PTH measurement and the additional parameter phosphorus (P), namely the measurement after six months, P(6), and the initial measurement, P(0). Informally, the solution states that with an increase of phosphorus goes along an increase of PTH too. More formally, the retrieved solution states:

If P(6) >= P(0)
then PTH(6) >= PTH(0)
else PTH(6) < (PTH(0)

So, here a complete restoration of the missing PTH value is not possible but just a binary solution that indicates the trend, where "1" stands for an increase and "0" for a decrease.

4.2.3 Case-Based Reasoning

In ISOR, cases are mainly used to explain further exceptional cases that do not fit the initial model. One such secondary application is the restoration of missing data. The solutions given by the medical expert are stored in the form of cases so that they can be retrieved for solving further missing data cases. Such a case stored in the case base has the following structure:

1. Name of the patient
2. Diagnosis
3. Therapy
4. Problem: missing value
5. Name of the parameter of the missing value
6. Measurement time point of the missing value
7. Formula of the solution (the description column of Table 3)
8. Reference to the internal implementation of the formula

9. Parameters used in the formula
10. Solution: Restored value
11. Type of solution (exact, estimated, or binary)

Since the number of stored cases is rather small (at most 150), retrieval is not crucial. The retrieval is performed by keywords. The four main keywords are: Problem code (here: "missing value"), diagnosis, therapy, and time period. As an additional keyword the parameter where the value is missing can be used. Solutions that are retrieved by using the additional keyword are domain dependent. They contain medical knowledge that has been provided by the medical expert. The domain independent solutions are retrieved by using just the four main keywords. The flow of restoring missing values in shown in Figure 5.

What happens when the retrieval provides more than one solution? Though only very few solutions are expected to be retrieved at the same time, only one solution should be selected. At first ISOR checks whether the required parameter values of the retrieved solutions are available. A solution is accepted if all required values are available. If more than one solution is accepted, the expert selects one of them. If no solution is accepted, ISOR attempts to apply the one with the fewest required parameter values.

Each type of solution has its specific adaptation. A numerical solution is just a result of a calculation according to a formula. This kind of adaptation is performed automatically. If all required parameter values are available, the calculation is carried out and the query case receives its numerical solution.

The second kind of adaptation modifies a restoration formula. This kind of adaptation can not be done entirely automatically but the expert has to be involved. When a (usually short) list of solutions is retrieved, ISOR at first checks whether all required values of the exact calculation formulae are available. If required parameter

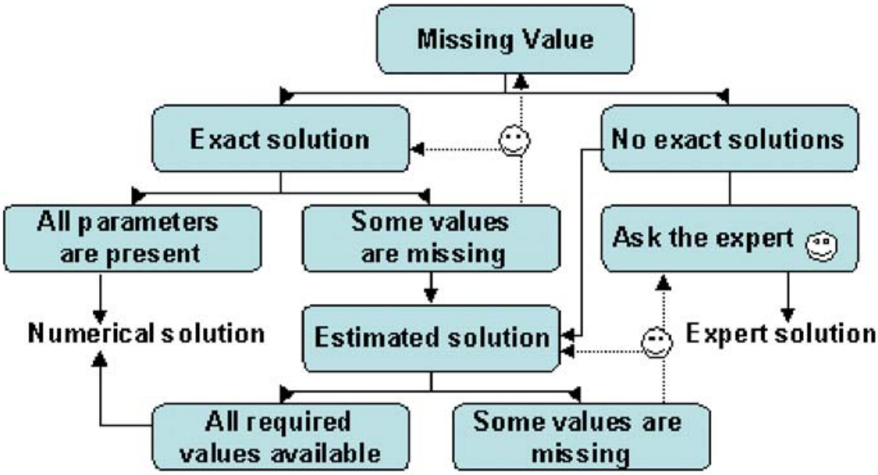


Fig. 5 Flow of restoration

values are not available, there are three alternatives to proceed. First, to find an exact solution formula where all required parameter values are available, second to find an estimation formula, and third to attempt to restore the required values too. Since for the third alternative there is the danger that this might lead to an endless loop, this process can be manually stopped by pressing a button in the dialogue menu. When for an estimated solution required values are missing, ISOR asks the expert.

The expert can suggest an exact or an estimated solution. Of course, such an expert solution also has to be checked for the availability of the required data. However, the expert can even provide just a numerical solution, a value to replace the missing data – with or without an explanation of this suggested value.

Furthermore, adaptation can be differentiated according to its domain dependency. Domain dependent adaptation rules have to be provided by the expert and they are only applicable to specific parameters. Domain independent adaptation uses general mathematical formulae that can be applied to many parameters. Two or more adaptation methods can be combined.

In ISOR a revision occurs. However, it is a rather simple one. It is not as sophisticated as, for example, the theoretically one described by Lieber [16]. Here, it is just an attempt to find better solutions. An exact solution is obviously better than an estimated one. So, if a value has been restored by estimation and later on (for a later case) the expert has provided an appropriate exact formula, this formula should be applied to the former case too. Some estimation rules are better than others. So it may happen that later on a more appropriate rule is incorporated in ISOR. In principle, the more new solution methods are included in ISOR, the more former already restored values can be revised.

Artificial cases. Since every piece of knowledge provided by a medical expert is supposed to be valuable, ISOR saves it for future use. If an expert solution cannot be used for adaptation for the query case (required values for this solution might be missing too), the expert user can generate an artificial case. In ISOR there exists a special dialogue menu to do this. Artificial cases have the same structure as real ones, and they are also stored in the case base.

5 Results

At first, we undertook some experiments to assess the quality of our restoration method, subsequently we attempted to restore the real missing data, and finally we set up a new model for the original hypothesis that actively participating in the fitness program improves the conditions of the patients.

5.1 Experimental Results

Since ISOR is a dialogue system and the solutions are generated within a conversation process with the user, the quality of the solutions not only depends on ISOR but also on the expert user.

Table 4 Summary of randomly deleted and restored values. Only the deleted values were attempted to restore, but not the really missing ones

Number of Parameters	112
Number of values	448
Number of really missing values	104
Number of randomly deleted values	97
Number of completely restored values	29
Number of estimated values	17
Number of partly restored values (binary)	13
Number of automatically restored values	34
Number of expert assistance	25
Number of values that could not be restored	38

Table 5 Closeness of the restored values. The numbers in brackets show the deviations on average in percentage

Deviation	Number of exactly restored values	Number of estimated values
$<$ 3 %	14 (2.2)	9 (1.8)
$<$ 5 %	13 (5.7)	5 (6.1)
$<$ 10 %	2 (8.5)	2 (7.4)
$>$ 10 %	0	1 (12.3)

To test the method a random set of parameter values was deleted from the observed data set. Subsequently, the method was applied and it was attempted to restore the deleted values - but not the ones actually missing!

Table 4 summarises how many deleted values could be restored. Since, for those 12 parameters that were only measured once and remain constant throughout, no values were deleted (and none of them are really missing), they are not considered in Table 4. More than half of the deleted values could be at least partly restored, nearly a third of the deleted values could be completely restored, about 58% of restoration occurred automatically. However, 39% of the deleted values could not be restored at all. The main reasons are that for some parameters no proper method is available and that specific additional parameter values are required that are sometimes also missing.

Another question concerns the quality of the restoration. That means how close are the restored values to the real values? We have to distinguish between exact, estimated and binary restored values. Just one of the 13 binary restored values was wrong. However, this mainly shows the "quality" of the expert user, who probably was rather cautious and made binary assessments only when he/she felt very sure. The deviation (percentage) between the restored values and the real ones is shown in Table 5. Concerning the two exactly restored values with more than 5% deviation, we consulted the expert user, who consequently altered one formula, which had been applied for both values. For the estimated values, it is conspicuous that for a few values the deviation is rather large. The probable reason is that general estimation methods have problems in discovering underlying patterns in certain cases. For example, with sequences such as 5, 7, 10 and so on, restoration can be quite straightforward. On the other hand, sequences like 5, 10, 3 can prove problematic.

5.2 Restoration of Real Missing Data and Setting up a New Model

As a consequence of the experimental results, we assumed that our method is not perfect but at least practical. So, we attempted to restore the real missing data. The result is shown in Table 6.

Table 6 Summary of missing and restored values

Number of Parameters	112
Number of values	448
Number of missing values	104
Number of completely restored values	37
Number of estimated values	24
Number of partly restored values (binary)	19
Number of automatically restored values	49
Number of expert assistance	31
Number of values that could not be restored	24

It is no surprise that more missing values could be restored (Table 6) than randomly deleted ones (see Table 2). As all restoration methods rely on other parameter values, the more parameter values you have, the better the chance of restoring missing values. In the experiment (Table 4) not just the randomly deleted values were missing but also the real missing ones.

After this restoration we return to the original problem, namely to set up a model for the hypothesis that actively participating in the fitness program improves the conditions of the patients (see section 2.1). Since many missing values have been restored, the expert user can choose other main factors to set up the model, including ones where many data had been missing previously. In fact, the expert user now chose a different third factor than before, namely PTH instead of WorkJ. The resulting strongest model is shown in Table 7.

Table 7 Results of Fisher's exact test, for $p < 0.0001$

Patient's physical condition	Active	Non-active	Fisher Exact p
Better	39	1	< 0.0001
Worse	11	21	

The result is obviously much better than the previous model (see Table 1 in section 2.1). However, since the missing data problem is not responsible for all exceptional cases, for this model some (eleven) exceptional cases still have to be explained.

6 Conclusion

In this chapter, it has been proposed to use CBR in ISOR to explain cases that do not fit a statistical model. Here one of the simplest statistical models was presented. However, it is relatively effective, because it demonstrates statistically significant dependencies. In our example, relating fitness activity to health improvement for dialysis patients, the model covers about two thirds of the patients, whereas the other third can be explained by applying CBR.

Since binary qualitative assessments (better or worse) were chosen, very small changes appear identical to very large ones. As a future step, it is intended to define these concepts more precisely, especially to introduce more assessments.

The presented method makes use of different sources of knowledge and information, including medical experts. This approach seems to be a very promising method to deal with a poorly structured database, with many missing data, and with situations where cases contain different sets of attributes.

Additionally, a method to restore missing values was developed. This method combines general domain independent techniques with expert knowledge, which is delivered as formulae for specific situations (treated as cases) and can be used for later similar situations too. The expert knowledge is gained within a conversational process between the medical expert, ISOR, and the system developer. Since the time of the expert is valuable, he/she is only consulted when absolutely necessary.

In ISOR, all main CBR steps are performed: retrieval, adaptation, and revision. Retrieval (of usually a list of solutions) occurs with the help of keywords. Adaptation (just like part of the restoration process of missing data) is an interactive process between ISOR, a medical expert, and the system developer. In contrast to many CBR systems, in ISOR revision plays an important role. The whole knowledge is contained in the case base, namely as solutions of former cases. No further knowledge base is required.

In principle, the active incorporation of a medical expert into the decision making process seems to be a promising idea. Already in our previous work [26], a successful Case-Based Reasoning system was developed that performed a dialog with a medical expert user to investigate therapy inefficacy.

Since CBR seems to be appropriate for medical applications (see section 2.1) and many medical CBR systems have already been developed, it makes sense to combine both ideas, namely to build systems that a both, case-oriented and dialog-oriented.

Acknowledgements. We thank Professor Alexander Rumyantsev from the Pavlov State Medical University for his close co-operation. Furthermore we thank Professor Aleksey

Smirnov, director of the Institute for Nephrology of St-Petersburg Medical University and Natalia Korosteleva, researcher at the same Institute, for collecting and managing the data.

References

1. Aamodt, A., Plaza, E.: Case-Based Reasoning: foundation issues. Methodological variation and system approaches. AI Commun. 7(1), 39–59 (1994)
2. Aha, D.W., McSherry, D., Yang, Q.: Advances in conversational Case-Based Reasoning. Knowledge Engineering Review 20, 247–254 (2005)
3. Arshadi, N., Jurisica, I.: Data Mining for Case-based Reasoning in high-dimensional biological domains. IEEE Transactions on Knowledge and Data Engineering 17(8), 1127–1137 (2005)
4. Bichindaritz, I., Kansu, E., Sullivan, K.M.: Case-based Reasoning in Care-Partner. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS, vol. 1488, pp. 334–379. Springer, Heidelberg (1998)
5. Broder, A.: Strategies for efficient incremental nearest neighbor search. Pattern Recognition 23, 171–178 (1990)
6. Corchado, J.M., Corchado, E.S., Aiken, J., Fyfe, C., Fernandez, F., Gonzalez, M.: Maximum likelihood Hebbian learning based retrieval method for CBR systems. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS, vol. 2689, pp. 107–121. Springer, Heidelberg (2003)
7. Davidson, A.M., Cameron, J.S., Grünfeld, J.-P. (eds.): Oxford Textbook of Nephrology, vol. 3. Oxford University Press, Oxford (2005)
8. Fleiss, J.: The design and analysis of clinical experiments. John Wiley & Sons, Chichester (1986)
9. Gediga, G., Düntsch, I.: Maximum Consistency of Incomplete Data via Non-Invasive Imputation. Artificial Intelligence Review 19(1), 93–107 (2003)
10. Gierl, L.: Klassifikationsverfahren in Expertensystemen für die Medizin. Mellen Univ. Press, Lewiston (1992)
11. Gierl, L., Bull, M., Schmidt, R.: CBR in Medicine. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.) Case-Based Reasoning Technology. LNCS, vol. 1400, pp. 273–297. Springer, Heidelberg (1998)
12. Hai, G.A.: Logic of diagnostic and decision making in clinical medicine. Politheknica publishing, St. Petersburg (2002)
13. Kendall, M.G., Stuart, A.: The advanced theory of statistics, 4th edn. Macmillan publishing, New York (1979)
14. Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann Publishers, San Mateo (1989)
15. Lenz, M., Auriol, E., Manago, M.: Diagnosis and decision support. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.) Case-Based Reasoning Technology. LNCS, vol. 1400, pp. 51–90. Springer, Heidelberg (1998)
16. Lieber, J.: Application of the revision theory to adaptation in Case-Based Reasoning: The conservative adaptation. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS, vol. 4626, pp. 239–253. Springer, Heidelberg (2007)
17. Little, R., Rubin, D.: Statistical analysis with missing data. John Wiley & Sons, Chichester (1987)
18. McSherry, D.: Interactive Case-Based Reasoning in sequential diagnosis. Applied Intelligence 14(1), 65–76 (2001)

19. Montai, S., Magni, P., Bellazzi, R., Larizza, C., Roudsari, C., Carsson, E.R.: Integration model-based decision support in multi-modal reasoning system for managing type 1 diabetic patients. Artificial Intelligence in Medicine 29(1-2), 131–151 (2003)
20. Nilsson, M., Sollenborn, N.: Advancements and trends in medical case-based reasoning: An overview of systems and system developments. In: Proceedings Seventeenth International Florida Artificial Intelligence Research Society Conference, pp. 178–183. AAAI Press, Menlo Park (2004)
21. Perner, P. (ed.): Case-Based Reasoning on Images and Signals. Springer, Berlin (2007)
22. Prentzas, J., Hatzilgeroudis, I.: Integrating Hybrid Rule-Based with Case-Based Reasoning. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS, vol. 2416, pp. 336–349. Springer, Heidelberg (2002)
23. Rezvani, S., Prasad, G.: A hybrid system with multivariate data validation and Case-based Reasoning for an efficient and realistic product formulation. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003, vol. 2689, pp. 465–478. Springer, Heidelberg (2003)
24. Schank, R.C., Leake, D.B.: Creativity and learning in a case-based explainer. Artificial Intelligence 40, 353–385 (1989)
25. Schmidt, R., Montani, S., Bellazzi, R., Portinale, L., Gierl, L.: Case-Based Reasoning for Medical Knowledge-based Systems. International Journal of Medical Informatics 64(2-3), 355–367 (2001)
26. Schmidt, R., Vorobieva, O.: Case-Based Reasoning Investigation of Therapy Inefficacy. Knowledge-Based Systems 19(5), 333–340 (2006)
27. Shuguang, L., Qing, J., George, C.: Combining case-based and model-based reasoning: a formal specification. In: Proc. APSEC 2000, p. 416 (2000)
28. Stottler, R.H., Henke, A.L., King, J.A.: Rapid retrieval algorithms for case-based reasoning. In: Proc of 11th Int Joint Conference on Artificial Intelligence, pp. 233–237. Morgan Kaufmann, San Mateo (1989)
29. Wilke, W., Smyth, B., Cunningham, P.: Using Configuration Techniques for Adaptation. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.) Case-Based Reasoning Technology. LNCS, vol. 1400, pp. 139–168. Springer, Heidelberg (1998)

Collaborative and Experience-Consistent Schemes of System Modelling in Computational Intelligence

Witold Pedrycz

Abstract. Computational Intelligence (CI) is commonly regarded as a synergistic framework within which one can analyze and design (synthesize) intelligent systems. The methodology of CI has been firmly established through the unified and highly collaborative usage of the underlying information technologies of fuzzy sets (and granular computing, in general), neural networks, and evolutionary optimization. It is the collaboration which makes the CI models highly versatile, computationally attractive and very much user-oriented. While this facet of functional collaboration between these three key information technologies has been broadly recognized and acknowledged within the research community, there is also another setting where the collaboration aspects start to play an important role. They are inherently associated with the nature of intelligent systems that quite often become distributed and whose interactions come with a suite of various mechanisms of collaboration. In this study, we focus on collaborative Computational Intelligence which dwells upon numerous forms of collaborative linkages in distributed intelligent systems. In the context of intelligent systems we are usually faced with various sources of data in terms of their quality, granularity and origin. We may encounter large quantities of numeric data coming from noisy sensors, linguistic findings conveyed by rules and associations and perceptions offered by human observers. Given the enormous diversity of the sources of data and knowledge the ensuing quality of data deserves careful attention. Knowledge reuse and knowledge sharing have been playing a vital role in the knowledge management which has become amplified over the time we encounter information systems of increasing complexity and of a distributed nature. The collaborative CI is aimed at the effective exchange of locally available knowledge. The exchange is commonly accomplished by interacting at the level of information granules rather than numeric quantitative. As a detailed

Witold Pedrycz
Department of Electrical & Computer Engineering, University of Alberta,
Edmonton AB T6R 2G7 Canada, and
System Research Institute,
Polish Academy of Sciences, Warsaw, Poland
e-mail: pedrycz@ee.ualberta.ca

C.L. Mumford and L.C. Jain (Eds.): Computational Intelligence, ISRL 1, pp. 697–723.
springerlink.com

case study we discuss experience–consistent modeling of CI constructs and raise an issue of knowledge reuse in the setting of constructs of Computational Intelligence. One could note that the knowledge-based component (viz. previously built CI constructs) can serve as a certain form of the regularization mechanism encountered quite often in various modeling platforms. The optimization procedure applied there helps us strike a sound balance between the data-driven and knowledge-driven evidence. We introduce a general scheme of optimization and show an effective way of reusing knowledge. In the sequel, we demonstrate the development details with regard to fuzzy rule-based models and neural networks.

1 Introductory Comments

Complex phenomena such as those encountered in human-centric systems , in which the human factor is predominant, are highly multifaceted. This concerns a panoply of economic and social systems which can be looked at and comprehended from various perspectives (points of view) and levels of abstraction. These phenomena are distributed and generate significant masses of data which become available locally with eventual restriction on their possible availability on a global basis. The global economy as a system is not described by a single model but its holistic view is formed by studying its behavior at more local and individually selected levels where building a model could be more feasible. Afterwards through interaction between the models and reconciliation of their findings a general global model is sought. The distributed and collaborative way of global model building becomes an interesting tendency that is worth adopting in the current practice of system modeling. Distributivity of the systems is a result of the existence of locally available data. Collaborative interaction supports a coherent formation of findings and facilitates the reconciliation of differences and reinforcement of some commonalities (general findings).

There are two fundamental dimensions of the overview perspective established for the phenomena under considerations (refer to Figure 1). First, a certain perspective arises on a basis of some features (attributes) of the phenomenon. Different subsets of features could offer complementary and equally relevant views of the system under discussion. The individual subsets of features could be disjoint. They can also overlap. Each of these subsets gives rise to various models describing the system from the different individual standpoint. These models, when combined together, are helpful in forming a global and comprehensive view at the phenomenon. The second coordinate is associated with the concept of cognitive perspective [8, 9, 10]. The most promising level of information granularity to be captured within the developed model is used to establish a suitable cognitive perspective.

The collaboration predominantly occurs at the level of information granules [14, 15] which can be represented as fuzzy sets, sets , rough sets and others. There are two interesting scenarios of substantial generality which will be discussed in detail. In the first one, whose essence is outlined in Figure 2(a), the development (reconciliation) of the information granules is realized by running any algorithm of information granulation which processes the locally available data while taking into

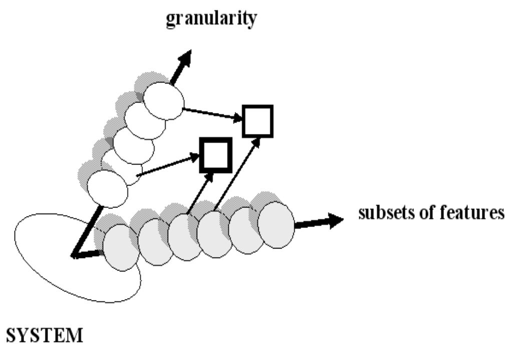


Fig. 1 Two fundamental dimensions of system modeling: through collections of features (attributes, variables) and by admitting a variable level of granularity and establishing a suitable cognitive perspective. Note two models (depicted as squares) are formed by considering specific subsets of features and selected information granularity

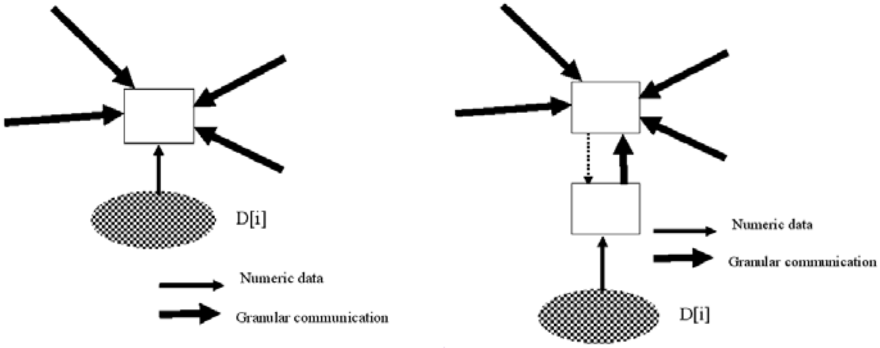


Fig. 2 Two general modes of reconciliation of information granules

account the description of information granules available at the collaborating data sites. At the algorithmic end, we envision a certain augmentation of the original objective function which is needed to support this form of collaboration.

The second general scheme of collaboration is realized by completion of clustering of granular findings coming from both the local data site and the remaining data sites, Figure 2 (b). The resulting information granules (which in this context might be referred to as meta-granules) are sent back to the local data site where they help in further navigation of the formation of the information granules. In this sense there is some feedback loop formed there being visualized by the solid and dotted lines.

Generally speaking, in fuzzy information processing, not too much has been said about its schemes of a distributed nature. While there has been a wealth of methodological and algorithmic developments in fuzzy modeling, the subject of distributed and collaborative fuzzy models has not been investigated in great detail. For instance, a lot has been said about rule-based fuzzy models of the form “if **x** is A_i then y =f_i(**x**, $\mathbf{a}_i$), i=1, 2, ...,c where A_i are fuzzy sets defined in the multidimensional input space and f_i denotes a local model endowed with some parameters ($\mathbf{a}_i$). What if we encounter individual data sites D[1], D[2], ..., D[P] for which such models have to be constructed? Not only do they have to be formed on a basis of locally available data D[ii], ii =1, 2, ..., P but they should collaborate, exchange their findings and reconcile eventual differences. In other words, the communication involves *knowledge* instead of *data*. In the communication scheme of this nature we witness an effect of knowledge sharing. Formally, the underlying knowledge being shared between the individual sites can be represented as **K**[ii]. For instance, for the rule based-systems, **K**[ii] = {A_i[ii], i=1, 2, ..., c}where Ai[ii] are the information granules (fuzzy sets) formed at D[ii]. In this way, the knowledge of these fuzzy sets is communicated to the other data sites. We may have another format of **K**[ii] being a more comprehensive version of knowledge sharing which concerns both the information granules and the local models, that is **K**[ii] = {A_i[ii], f_i[ii], $\mathbf{a}_i$[ii]}.

The study brings forward a number of developments which form a conceptual and algorithmic framework of collaborative Computational Intelligence. In Section 2, we present the fundamentals of collaborative clustering where we show how information granules – fuzzy sets – emerge as a result of knowledge sharing. Then in Sections 3 and 4 we present the algorithmic aspects of the method by showing a general flow of computing and the pertinent computing details. Hierarchies of clusters are introduced in Section 5. Experience-consistent fuzzy modeling brings along a new concept and it is presented in the context of rule-based fuzzy models and neural networks. Section 6 is devoted to experience-consistent fuzzy models whose development embraces experimental data and some previous experience - knowledge hints coming in the form of previous models. In the sequel, the concept of experience consistency is further discussed in application to the design of radial basis function neural networks (Section 7). Conclusions are presented in Section 8.

To focus our discussion and discuss an algorithmic setup in a tangible fashion, we consider information granules that are constructed through fuzzy clustering, and Fuzzy C-Means (FCM), in particular [1, 2, 3, 4, 11, 13]. We assume that all data sites D[1], D[2], . .. , D[P] comprise data positioned in the same n-dimensional feature space $\mathbf{R}^n$.

2 Collaborative Clustering

The communication of knowledge involves a structure **K**[ii] which embraces a collection of information granules – fuzzy clusters. Considering that such clusters have been constructed with the use of the FCM algorithm, they are fully characterized in terms of prototypes and partition matrices. As a matter of fact, these two characterizations are equivalent in the sense that for the given data {$\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_N$} the prototypes are expressed by means of the partition matrix while the partition matrix comes with the entries whose computing involves the knowledge of the prototypes. The prototypes and partition matrices are the two possible communication vehicles between the data sites. Given the fact that the data sites concern different data sets

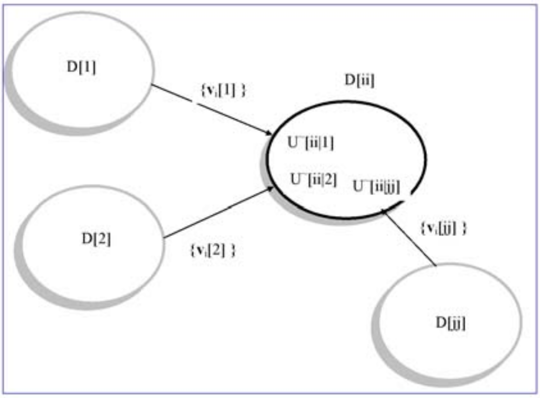


Fig. 3 Data sites and communication realized through passing prototypes and the consecutive generation of the induced partition matrices U~[ii|jj]

of possibly different cardinalities, sharing knowledge about the partition matrices is not feasible at all. The prototypes, on the other hand, form a viable alternative. Communicating a limited number of prototypes is also highly attractive since in this manner no significant communication overhead builds up. As the FCM optimization focuses on the partition matrices as one of its components to be adjusted throughout collaboration, we introduce a concept of so-called *induced* partition matrices. Consider the ii-th data site. The prototypes produced at the jj-th data site $\mathbf{v}_1$[jj], $\mathbf{v}_2$[jj],..., $\mathbf{v}_c$[jj] are communicated to the ii-th data site. Given this collection of the prototypes, we induce a partition matrix over the data site D[ii]. Denote it by U~[ii|jj] where the two indexes (ii and jj) point at data sites taking part in this interaction. Its entries are determined in a standard way encountered in FCM computing [1], that is

$$u^{\sim}{}_{ik}[ii|jj] = \frac{1}{\sum_{j=1}^{c}\left(\frac{||\mathbf{x}_k[ii]-\mathbf{v}_i[jj]||}{||\mathbf{x}_k[ii]-\mathbf{v}_j[jj]|}\right)^2} \tag{1}$$

i=1, 2,..., c; k=1, 2,...,N[ii] and $\mathbf{x}_k$ ∈D[ii]. Refer also to Figure 3 which highlights the essence of this mechanism of the collaboration by visualizing a way in which the communication links have been established.

Proceeding with all other data sites, D[1], ..., D[ii-1], D[ii+1],..., D[P], we end up with P-1 induced partition matrices , U~[ii|1], U~[ii|2],...., U~[ii|ii-1], U~[ii|ii+1],...., U~[ii|P]. The minimization of differences between the U[ii] and U~[ii|jj] is used to establish some collaborative activities occurring between the data sites. At the ii-th site, the clustering is guided by the augmented objective function assuming the following form

$$Q[ii] = \sum_{k=1}^{N[ii]}\sum_{i=1}^{c} u_{ik}^2[ii]\,\|x_k - v_i\|^2 \quad + \quad \beta \sum_{\substack{jj=1\\ jj\neq ii}}^{P}\sum_{k=1}^{N[ii]}\sum_{i=1}^{c}(u_{ik}[ii] - u_{ik}^{\sim}[ii|jj])^2 d_{ik}^2 \tag{2}$$

where β is a certain nonnegative number (scaling coefficient). The objective function Q[ii] consists of two components. The first one is nothing but a standard sum of weighted distances between the patterns in the standard FCM being applied to D[ii] with the fuzzification coefficient m =2. The second component reflects an impact coming from the structures formed at all remaining data sites. The distance between the optimized partition matrix and the induced partition matrices is to be minimized – this requirement is captured by this part of the objective function (2). The scaling coefficient β strikes a sound balance between the optimization guided by the structure in D[ii] and the already developed structures available at the remaining sites. The value of β implies a certain level of intensity of collaboration; the higher its value, the stronger the collaboration. For $\beta = 0$ no collaboration occurs and the problem reduces to the collection of "P" independently run clustering tasks being exclusively confined to the corresponding data sites.

In a nutshell, the problem of collaborative clustering can be formulated as follows:

> Given a finite number of disjoint data sites with patterns defined in the same feature space, develop a scheme of collective development and reconciliation of a fundamental cluster structure across the sites that it is based upon exchange and communication of local findings where the communication needs to be realized at some level of information *granularity*. The development of the structures at the local level exploits the communicated findings in an *active* manner through minimization of the corresponding objective function augmented by the structural findings developed outside the individual data site. We also allow for retention of key individual (specific) findings that are essential (unique) for the corresponding data site.

Schematically, we portray the essence of the collaborative clustering as presented in Figure 4, which stresses an act of balancing between collaborative activities occurring between the data sites and reflecting *global* and common characteristics of all data and the crucial findings implied by the *locally* available data.

Alluding to Figure 4, we can offer another important and visible category of applications which deal with wireless sensor networks. In such networks, we envision a collection of randomly scattered sensors whose communication is established on an ad hoc basis. Each node (sensor) collects the data available in its neighborhood and realizes their processing leading to the determination of the *local* characteristics of data (say, formulated as a collection of clusters being observed at this particular local level of the given sensor). At the same time it is recognized that the local processing could benefit from some collective activities established between the sensors. This need for a *global* and collective style of processing is motivated by a limited amount of data available locally and a need to establish a global view of the data collected by the overall network. Each sensor formulates a very limited and localized perception of the environment that has to be augmented by local findings formed by other sensors.

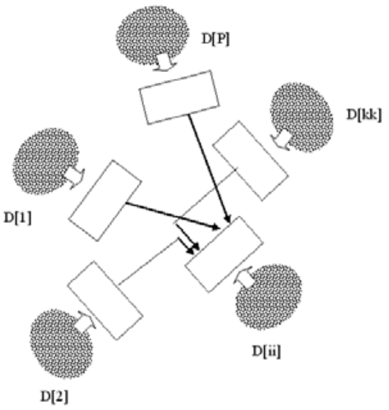


Fig. 4 The essence of collaborative clustering in which we aim at striking a sound balance between local findings (produced at the level of locally available data) and the findings coming from other data sites (sensors) building some global characterization of data. Shown are only communication links between data site D[ii] and all other data sites

3 The General Flow of Collaborative Processing

The essence of collaborative clustering pertains to the development of structures at individual data sites on the basis of effective communication of the findings obtained at the level of the individual data sites. There are two phases, namely an optimization of the structures at the individual sites and an interaction between them when exchanging the findings. They intertwine so that these two phases occur in a fixed sequence. Initially, the FCM algorithm is run independently at each data site (this happens without any communication). After FCM has terminated at each site, processing stops and the data sites communicate their findings. As already stressed, this communication needs to be realized at some level of information granularity. The effectiveness of the interaction depends on the way in which one data site "talks" to others in terms of what has been discovered so far. We discuss the pertinent details later. Once communication has been established and the nodes are informed about structural findings at other sites, each site proceeds with its optimization pursuits by focusing on the local data while at this point taking into consideration the findings communicated by other data sites. These optimization processes are run independently from each other. Once all of them have declared termination of computing, they are ready to engage in the communication phase. Again they communicate the findings and set up new conditions for the next phase of the FCM optimization. The pair of optimization and communication processes is referred to as a collaboration phase. The overall collaboration takes a finite number of collaboration phases (phases, for short), which terminates once no further significant change in the revealed structure is reported.

As has become clear from this high-end description of the collaboration, there are two important components crucial to the overall process. First, we have to specify a way of communicating and representing findings at some level of granularity (let us recall that we are not allowed to communicate at the level of individual data but have to establish communication at the higher level of abstraction by engaging the exchange of the granular constructs). Second, we have to come up with an augmented objective function whose minimization embraces both the structures at the local level of the individual data sites and reconciles them with the structures communicated by other data sites.

4 Algorithmic Aspects of Collaborative Clustering

In what follows we discuss several main algorithmic issues of collaborative clustering.

4.1 The Computing Scheme

The objective function Q[ii] expressed by Equation (2) consists of two components. Its minimization is quite standard. Making use of Lagrange multipliers we derive detailed formulas for the partition matrix and a set of prototypes; both of them are determined in an iterative fashion. The overall scheme of the collaborative clustering is outlined as follows

Given: data sites D[1], D[2], …, D[P]
Choose the number of clusters (c) to be looked for in the collaborative clustering, set up some termination criterion of the FCM, and establish a level of collaboration (interaction) by choosing some nonnegative value of β.

Initial phase Carry out clustering (FCM) for each data site producing a collection of prototypes $\{\mathbf{v}_i[\text{ii}]\}$, i=1,2,…,c for each data site.

Collaboration
Iterate {successive phases of collaboration}

Communicate the results about the structure determined at each data site.
For each data site (ii)
{
Minimize performance index (2) at each data site by iteratively proceeding with the calculations of the partition matrix and the prototypes, that is

$$u_{rs}[ii] = \frac{1}{\sum\limits_{j=1}^{c} \frac{d_{rs}^2}{d_{js}^2}} \left[1 - \sum_{j=1}^{c} \frac{\beta \sum\limits_{\substack{jj=1 \\ jj\neq ii}}^{P} u^{\sim}_{js}[ii|jj]}{[1+\beta(P-1)]} \right] + \frac{\beta \sum\limits_{\substack{jj=1 \\ jj\neq ii}}^{P} u^{\sim}_{rs}[ii|jj]}{[1+\beta(P-1)]} \qquad (3)$$

and

$$v_{rt}[ii] = \frac{\sum\limits_{k=1}^{N[ii]} u_{rk}^2[ii]x_{kt} + \beta \sum\limits_{\substack{jj=1 \\ jj\neq ii}}^{P} \sum\limits_{k=1}^{N[ii]} (u_{rk}[ii] - u_{rk}^{\sim}[ii|jj])^2 x_{kt}}{\sum\limits_{k=1}^{N[ii]} u_{rk}^2[ii]+\beta \sum\limits_{\substack{jj=1 \\ jj\neq ii}}^{P} \sum\limits_{k=1}^{N[ii]} (u_{rk}[ii] - u_{rk}^{\sim}[ii|jj])^2} \tag{4}$$

r=1,2,. . . ,c; t =1, 2, . . . , n; s =1, 2, . . . , N[ii]

} for data site

until termination condition of the collaboration activities has been satisfied.

4.2 Evaluation of the Quality of Collaboration: Striking a Sound Compromise between Global and Local Characteristics of Data

The evaluation of the quality of the results of collaboration between the data sites requires a careful assessment. As there are partition matrices associated with each of the D[ii]'s, one could think of computing distance between them and treat it as a measure of quality of the ongoing process. While the idea sounds convincing, its realization requires more attention. We should stress the fact that a direct comparison of two partition matrices could not be feasible [5, 12] as we may not have a direct correspondence between their rows (respective clusters). This is a well-known problem identified in the literature, cf. [6]. To get around this shortcoming, we use the concept of proximity and proximity matrix induced by a given partition matrix. Let us recall that for any partition matrix U = [u$_{ik}$], i=1,2,.., c, k=1,2, . . . ,m, an induced proximity matrix, that is Prox = [prox(k,l)], k, l=1,2,. . . ,m, comes with entries which satisfy the following properties

(a) symmetry prox (k_1,k_2) = prox(k_2,k_1)

(b) reflexitivity prox(k_1,k_1) =1.0

Interestingly enough, here we do not require transitivity (which, albeit nice to have, is always difficult to achieve in practice). The proximity values are based on the corresponding membership degrees occurring in the partition matrix

$$prox(k_1,k_2) = \sum_{i=1}^{c} min(u_{ik_1},u_{ik_2}) \tag{5}$$

It is worth noting that the proximity matrix is more abstract in this form than the original partition matrix it is based upon. It "abstracts" the clusters themselves and this is what we really need in this construct. Given the proximity matrix, we cannot "retrieve" the original entries of the partition matrix it was generated from.

Let us consider now the ii-th data site with its partition matrix U[ii] and the induced partition matrices $U^{\sim}[ii|jj]$, jj =1, 2, ..., ii-1, ii+1, ... , P. To quantify the consistency between the structure revealed at the ii-th data site with those existing at remaining sites by computing the following expression

$$W[ii] = \frac{1}{(N^2[ii]/2)} \sum_{\substack{jj=1 \\ jj\neq ii}}^{P} ||Prox(U[ii]) - Prox(U^{\sim}[ii|jj])|| \tag{6}$$

More specifically, we consider that the distance between the corresponding proximity matrices is realized in the form of the Hamming distance. In other words, we have

$$||Prox(U[ii]) - Prox(U^{\sim}[ii|jj])|| = \sum_{k_1=1}^{N[ii]} \sum_{k_2>k_1}^{N[ii]} |prox(k_1,k_2)[ii] - prox(k_1,k_2)^{\sim}[ii|jj]| \tag{7}$$

where prox(k_1,k_2)[ii] denotes the (k_1, k_2)- entry of the proximity matrix U[ii]. Similarly, Prox(k_1,k_2)$^{\sim}$[ii|jj] is the corresponding (k_1, k_2) entry of the proximity matrix produced by the induced partition matrix U$^{\sim}$[ii|jj]. In a nutshell, rather than working at the level of comparing the individual partition matrices (which requires knowledge of the explicit correspondence between the rows of the partition matrices), we generate their corresponding proximity matrices that allows us to carry out comparison at this more abstract level. Next summing up the values of W[ii] over all data sites, we arrive at the global level of consistency of the structure discovered collectively through the collaboration

$$W = W[1] + W[2] + \ldots + W[P] \tag{8}$$

The lower the value of W, the higher is the consistency between the “P” structures. Likewise the values of W being reported during the successive phases of the collaboration can serve as a sound indicator as to the progress and quality of the collaborative process and serve as a suitable termination criterion. In particular, when tracing the successive values of W, one could stop the collaboration once no further changes in the values of W are reported. The use of the above consistency measure is also essential when gauging the intensity of collaboration and adjusting its level through changes of β. Let us recall that this parameter shows up in the minimized objective function and shows how much other data sites impact the formation of the clusters at the given site. Higher values of β imply stronger collaborative linkages established between the sites. By reporting the values of W treated as a function of β, that is W =W(β), we can experimentally optimize the intensity of collaboration. One may anticipate that while for low values of β no collaboration occurs and the values of W tend to be high, large values of β might lead to competition and subsequently the values of W(β) may tend to be high. Under some conditions, no convergence of the collaboration process could be reported. There might be some regions of optimal values of β. Obviously, the optimal level (intensity) of collaboration depends upon a number of parameters of the collaborative clustering, in particular the number of clusters and the number of data sites involved in the collaboration. It could also depend upon the data themselves.

4.3 *Fuzzy Sets of Type-2 in the Quantification of the Effect of Collaboration*

Along with the ongoing collaboration, it is also advantageous to assess the quality of the results by evaluating their consistency and expressing a level of differences. Here the quantification of results completed in terms of type-2 fuzzy sets constitutes an interesting alternative or prototypes being treated as granular constructs, which is fuzzy sets rather than plain numeric entities. Recall that type-2 fuzzy sets are granular constructs that generalize fuzzy sets in the sense that their membership functions do no assume numeric membership grades but instead of them we encounter fuzzy sets defined in the unit interval. Interestingly, type-2 fuzzy sets have been discussed in various settings however very little was said about a determination of their membership functions. In collaborative clustering we estimate the membership function on a basis of a collection of membership grades available in different partition matrices. To be more specific, let us revisit what becomes known about cluster membership of pattern **x** in D[ii] given the available results of collaborative clustering. The membership in the i-th cluster is computed using the prototypes of D[ii] and is denoted as u = u_i. The prototypes optimized for the jj-th data site, jj =1, 2, ..., ii-1, ii+1, ..., P give rise to the membership of **x** to the same i-th cluster. Denote them by z_1, z_2, ..., z_{P-1}. All in all, we obtain a collection of membership grades which are now captured in a form of type-2 fuzzy set. The corresponding membership function is determined by solving a certain optimization problem [10][11][12] which realizes an idea which could be referred to as a principle of *justifiable granularity*. We consider triangular fuzzy set as one of the simplest versions of the membership functions. It is also legitimate in the context of this application given that we operate in presence of limited experimental evidence. The modal value of the fuzzy set is the membership value obtained with the use of the prototypes present at D is equal to "u". Consider now the values of z_i that are lower than u, $z_i < u$. We use them in the formation of the left-hand side of the linear portion of the membership function, refer to Figure 5.

There are two fundamental requirements guiding the design of the fuzzy set, namely

(a) maximize the experimental evidence of the fuzzy set; this implies that we tend to "cover" as many numeric data as possible, viz. the coverage has to be made as high as possible. Graphically, in the optimization of this requirement, we rotate the linear segment up (clockwise) as illustrated in Figure 5. Normally, the sum of the membership grades A(z_i), $\sum_i A(z_i)$where A is the linear membership function to be optimized with respect to its slope and z_i is located to the left to the modal value (u) has to be maximized

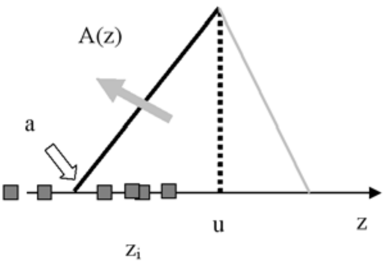


Fig. 5 Computation of a membership function of a fuzzy set of type-2; note that in order to maximize the performance index, we rotate the linear segment of the membership function around the modal value of the fuzzy set. Small dark boxes denote available experimental data. The same estimation procedure applies to the right-hand side of the fuzzy set

(b) Simultaneously, we would like to make the fuzzy set as specific as possible so that is comes with some well defined semantics. This requirement is met by making the support of A as small as possible, that is $\min_a|u - a|$

To accommodate the two conflicting requirements, we have to combine these two constraints (a) and (b) into the form of a single scalar index which in turn becomes maximized. Two alternatives could be sought, say

$$max_{a \neq u} \frac{\sum\limits_i A(z_i)}{|u-a|} \tag{9}$$

or

$$\sum_i (1 - A(z_i))(u - a) \tag{10}$$

The linearly decreasing portion of the membership function positioned at the right-hand side of the modal value (u) is optimized in the same manner. We exclude a trivial solution of a = u in which case the fuzzy set of type-2 collapses to a type-1 fuzzy set (with numeric values of membership function). We use this construct in the formation of granular prototypes and fuzzy sets of type-2.

4.4 Collaborative Clustering in Presence of Different Levels of Information Granularity

The method presented so far was quite restrictive in the sense that we assumed that the number of clusters at each of collaborating data sites is the same. While this could still be a viable alternative assuming that all the collaborative parties agree in advance on the level of granularity they are interested in to consider, in general this assumption could be considered quite restrictive and not realistic. A far more flexible scenario is the one in which each party considers its own number of clusters (which could be quite legitimate considering that data's structure could vary from site to site and one may consider variable levels of information granularity).

The choice of the number of clusters at each data site is beyond this study as this topic is well covered in the existing literature and supported by various algorithmic means including an extensive suite of cluster validity indexes. Given this, the algorithmic settings discussed so far have to be augmented. The major step would be to present information granules at each data site at the level of granularity that has been accepted before collaboration. There are several possible ways of doing this. Here we consider the one which uses clusters of the prototypes. Consider the ii-th data site. Before each phase of collaboration, we cluster the prototypes of this data site $\{\mathbf{v}_i[\mathrm{ii}]\}$, i=1, 2, ..., c[ii] and the prototypes communicated from all remaining data sites, $\{\mathbf{v}_i[\mathrm{jj}]\}$, i=1, 2, ..., c[jj], jj=1, 2, ii-1, ii+1,...,P. The number of clusters is the same as the number of clusters at this data site. The results are denoted by $\mathbf{v}_i^{\sim}$ i=1, 2,..., c[ii]. These new prototypes are used in the next steps of the collaborative clustering. More specifically, the minimized objective function comes in the form

$$Q[ii] = \sum_{i,k} u_{ik}^2[ii]\,||x_k - v_i[ii]||^2 + \quad \beta \sum_{i=1}^{c[ii]} u_{ik}^2[ii]\,||v_i[ii] - v_i^{\sim}[ii]||^2 \tag{11}$$

The overall flow of processing is realized in the following manner

Given: data sites D[1], D[2], ..., D[P] with different structures

Select a number of clusters (c[ii]) for each data site, set up some termination criterion and establish a level of collaboration (interaction) by choosing some non-negative value of β.

Initial phase

Carry out clustering (FCM) for each data site producing a collection of prototypes $\{\mathbf{v}_i[\mathrm{ii}]\}$, i=1,2,...,c[ii] for each data site.

Collaboration

Iterate {successive phases of collaboration}

Communicate the results about the structure determined at each data site.

For each data site (ii)

{

Collect all prototypes from other sites at data site (ii) and run FCM on that collection of all prototypes by selecting the same number clusters at that site to generate new prototypes $\mathbf{v}^{\sim}$[ii]. Minimize performance index (11) at each data site by iteratively proceeding with the iterative calculations of the partition matrix and the prototypes, that is

$$u_{rs}[ii] = \frac{1}{\sum_{j=1}^{c[ii]} \frac{||x_s - v_r[ii]||^2 + \beta\,||v_r[ii] - v_r^{\sim}[ii]||^2}{\left|\left|x_s - v_j[ii]\right|\right|^2 + \beta \left|\left|v_j[ii] - v_j^{\sim}[ii]\right|\right|^2}} \tag{12}$$

and

$$v_{rt}[ii] = \frac{\sum_{k=1}^{N} u_{rk}^2[ii]x_{kt} + \beta \sum_{k=1}^{N} u_{rk}^2[ii]v_{rt}^{\sim}[ii]}{\sum_{k=1}^{N} u_{rk}^2[ii]\,(1+\beta)} \tag{13}$$

r=1,2,. . . ,c[ii]; t =1, 2, . . . , n; s =1, 2, . . . , N

} for the data site

until termination condition of the collaboration activities has been satisfied.

The quality of collaboration is optimized by choosing a suitable value of β (which minimizes the performance index W given by Equation (8)).

5 Hierarchical Clusters of Clusters

In the previous collaboration strategy, we have assumed that the collaborating data sites exchange their findings (prototypes) which have been produced at the same level of granularity (there is the same number of clusters c[ii] across all collaborating parties). One can envision a different architecture and the underlying strategy of reconciling findings at the local level. This brings the concept of *clusters of clusters*. The essence of the method is that the structural findings formed at the lowest level are reconciled in the form of structure that is common to all local data sites. The prototypes at each D[ii] are considered together and clustered into "cc" clusters formed at the higher level. In the sequel, the resulting partition matrix is used to convey information about the behavior of the original prototypes when being confronted with structural findings (prototypes) at other data sites. More specifically, using the partition matrix U formed at the higher level of this hierarchy, we form some relevancy index γ(U) to quantify the impact on any of the prototypes coming from the data site. The index which applies to each column of U associates the ith prototype at data site D[ii] with γ_i(U)[ii] which articulates how much identity this prototypes retains when confronted with the data structure obtained at other data sites. The index is included in the modified objective function used to cluster data at the ii-th data site

$$Q = \sum_{i=1}^{c[ii]} \sum_{\mathbf{x}_k \in \mathbf{X}[ii]} \gamma_i(U)[ii]||\mathbf{x}_k - \mathbf{v}_i[ii]||^2 \tag{14}$$

The formation of the clusters of clusters is an interactive process: we start with the development of structure individually at D[ii], cluster the obtained prototypes and use the relevancy index to minimize the modified objective function as shown above. The clusters formed in this way are again clustered at the higher level of the hierarchy. This leads to new values of the relevance index and the process iterates until it stabilizes. The number of clusters "cc" assumed at the higher level plays an important role as a measure to express the intensity of reconciliation of the individual findings. Strong interaction becomes realized when we consider a few clusters. In the case when cc = c[1] + c[2]+... + c[P] there is no interaction at all (each prototypes retains its identity) and the values of γ_i(U)[ii] are all equal to 1 not affecting the form of the objective function and thus not changing the prototypes. The strength of the structural interaction controlled by the values of the number of clusters "cc" may affect the dynamics of collaboration with the likelihood that its lower values associated with stronger collaboration may imply eventual instability.

6 Experience-Consistent Fuzzy Modeling

Fuzzy rule-based models [7, 10, 15] and fuzzy modeling play a predominant role in system modeling realized in the context of fuzzy sets. The most recent studies are a genuine testimony to the wealth of approaches and new computational pursuits in this area. As usual in system modeling [10] including the development of fuzzy rule-based systems, we rely on an intensive and prudent usage of experimental data. We exploit the existing data in order to establish a structure of the respective model and estimate its parameters. With regard to the character of the usage of data, we encounter several fundamental problems that require careful attention. Generalization capabilities of the models rely in a direct way on the characteristics of data (in particular their representative capabilities with respect to the problem at hand) and the nature of the model itself. The characteristics of data deserve particular attention in case we encounter small data sets which could be also heavily affected by noise. The models developed on the basis of a limited and noisy data set typically exhibit low prediction capabilities. A certain alleviation of the problem of this nature could be realized by contemplating reliance on other sources of knowledge about the system to be modeled where they might have been acquired in the past. They are not necessarily data themselves (whose accessibility could be limited to various reasons) but could be available in the format of the parameters of the models. In the anticipated modeling scenario, it becomes advantageous not only to consider currently available data but also actively exploit previously obtained findings. Such observations bring us to the following formulation of the problem:

> Given some experimental data, construct a model which is *consistent* with the findings (models) produced for some previously available data. Owing to the existing requirements such as data privacy or data security of data as well as some other technical limitations, access to these previous data is not available. Instead we can take advantage of the knowledge coming in the form of the parameters of the existing models.

Considering the need to achieve a certain desired consistency of the proposed model with the previous findings, we refer to the development of such models as *experience-based* or *experience-consistent* fuzzy modeling.

When dealing with experience-consistent models, we may encounter a number of essential constraints which imply a way in which the underlying processing can be realized. For instance, it is common that the currently available data are quite limited in terms of its size (which implies a limited evidence of the data set) while the previously available data sets could be substantially larger meaning that relying on the models formed in the past could be beneficial for the development of the current model. There is also another reason in which the experience –driven component plays a pivotal role. The data set D could be quite small and affected by a high level of noise – in this case it becomes highly legitimate to seriously consider any additional experimental evidence available around.

In the realization of consistent-oriented modeling, we consider the following scenario. Given is a data set D using which we intend to construct a fuzzy rule-based model. There is a collection of data sets D_1, D_2, ..., D_P. For each of them developed is an individual fuzzy model. Those local models are available when seeking consistency with the fuzzy models formed for D_{ii}, ii=1, 2, ..., P. At the same time, it is worth stressing that the data sets themselves are not available to any processing and modeling realized at the level of D.

The underlying architectural details of the rule-based model considered in this study are as follows. For each data site D and D_{ii}, we consider the rules with local regression models assuming the form

Data D

$$\text{-if } \mathbf{x} \text{ is } B_i \text{ then } y = \mathbf{a}_i^T \mathbf{x} \tag{15}$$

where $\mathbf{x} \in \mathbf{R}^{n+1}$ and B_i are fuzzy sets defined in the n-dimensional input space, i=1, 2,..., c. The local regression model standing in the i-th rule is a linear regression function described by a certain vector of parameters $\mathbf{a}_i$. More specifically, the n-dimensional vector of the original input variables is augmented by a constant input so we have $\mathbf{x} = [x_1\ x_2\ \dots\ x_n\ 1]^T$ and $\mathbf{a} = [a_1\ a_2\ \dots\ a_n\ a_0]^T$ where a_0 stands for a bias term that translates the original hyperplane.

The same number of rules (c) is encountered at all other data sites, D_1, D_2, ..., D_P. The format of the rules is the same as for D, that is for the ii-th data sited D_{ii} we have

$$\text{-if } \mathbf{x} \text{ is } B_i[ii] \text{ then } y = \mathbf{a}_i[ii]^T \mathbf{x} \tag{16}$$

As before the fuzzy sets in the condition part of the i-th rule are denoted by B_i[ii] while the parameters of the local model are denoted by $\mathbf{a}_i$[ii]. The index in the square brackets refers to the specific data site, that is D_{ii} for $\mathbf{a}_i$[ii].

Alluding to the format of the data at D, it comes in the form of input – output pairs $(\mathbf{x}_k, y_k)$, k=1, 2,..., N which are used to carry out learning in a supervised mode. The previously collected data sets denoted by D_1, D_2, ..., D_P consists of N_1, N_2, and N_P data points. We assume that due to some technical and non-technical reasons, the data available at D_j cannot be shared with D. However, the communication between the data sites can be realized at a higher conceptual level such as those involving the parameters of the fuzzy models.

6.1 The Consistency-Based Optimization of Local Regression Models

To make the ensuing formulas concise, we use a shorthand notation FM, FM[1], FM[2], . . . , FM[P] to denote rule-based models pertaining to data D, D[1],. . . etc.

As usual the optimal parameters of the local models occurring in the conclusions of the rules are chosen in such a way so that they minimize the sum of squared errors

$$Q = \frac{1}{N} \sum_{\substack{x_k \in D \\ y_k \in D}} (FM(\mathbf{x}_k) - y_k)^2 \tag{17}$$

For given fuzzy sets of conditions, the determination of the parameters of the linear models is standard and well documented in the literature. Considering the form of the rule-based system, the output of the fuzzy model is determined as a weighted combination of the local models with the weights being the levels of activation of the individual rules. More specifically we have

$$\hat{y}_k = \sum_{i=1}^{c} u_i(\mathbf{x}_k)\mathbf{a}_i^T \mathbf{x}_k \tag{18}$$

where u$_{ik}$= u$_i$($\mathbf{x}_k$) is a membership degree of the k-th data $\mathbf{x}_k$ to the i-th cluster being computed on a basis of the already determined prototypes in the input space. In a nutshell Equation (18) comes as a convex combination of the local models which aggregates the local models by taking advantage of the weight factors expressing a contribution of each model based upon the activation reported in the input space.

The essence of the consistency-driven modeling is to form local regression models occurring in the conclusions of the rules on a basis of data D while at the same time making the model perform in a consistent manner (viz. close enough) to the rule-based model formed for the respective D$_i$'s. The following performance index strikes a sound balance between the model formed exclusively on a basis of data D and the consistency of the model with the results produced by the models formed on a basis of some other data sites D$_i$'s, that FM[j]($\mathbf{x}_k$)

$$V = \sum_{\substack{x_k \in D \\ y_k \in D}} (FM(\mathbf{x}_k) - y_k)^2 + \alpha \sum_{j=1}^{P} \sum_{\substack{x_k \in D \\ y_k \in D}} (FM(\mathbf{x}_k) - FM[j](\mathbf{x}_k))^2 \tag{19}$$

The calculations of FM[j]($\mathbf{x}_k$) for some $\mathbf{x}_k$ in D require some words of explanation. The model is communicated to D by transferring the prototypes of the clusters (fuzzy sets) and the coefficients of the linear models standing in the conclusions of the rules refer to Figure 6.

When used at D, the prototypes $\mathbf{v}_i$[j], i=1, 2,. . . ,c give rise to an induced partition matrix in which the k-th column (for data $\mathbf{x}_k$) assumes the following membership

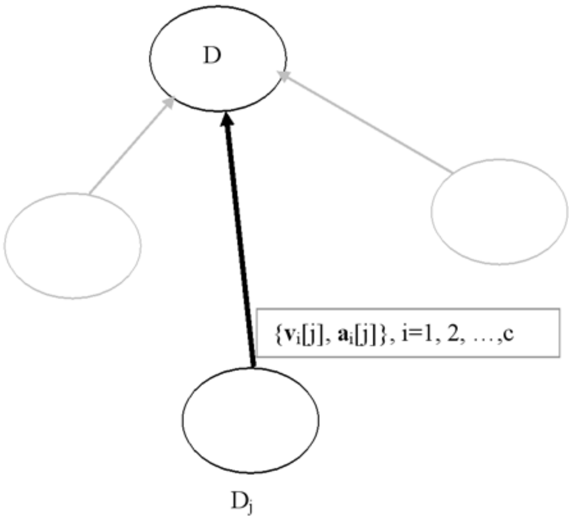


Fig. 6 Communication between D and D$_j$ realized by transferring parameters of the rule-based model available at individual data sites D$_j$

values w$_i(\mathbf{x}_k)$ computed in the standard manner as being encountered when running the FCM algorithm, that is

$$w_i(\mathbf{x}_k)[j] = \frac{1}{\sum_{l=1}^{c}\left(\frac{\mathbf{x}_k-\mathbf{v}_i[j]}{\mathbf{x}_k-\mathbf{v}_l[j]}\right)^{1/\text{m-2}}} \tag{20}$$

The transferred parameters of the local models obtained at the j-th data site produce the output of the model FM[j]($\mathbf{x}_k$) obtained at D as a weighted sum of the form

$$FM[j](\mathbf{x}_k) = \sum_{i=1}^{c} w_i(\mathbf{x}_k)[j]a_i^T(j)\mathbf{x}_k \tag{21}$$

where $\mathbf{x}_k \in$D.

The minimization of the performance index V for some predefined value of α leads to the optimal vectors of the parameters of the linear models $\mathbf{a}_i$(opt), i=1, 2,…, c which is reflective of the process of satisfying the consistency constraints. After some algebra, the final result comes in the form

$$\mathbf{a}_{opt} = \frac{1}{\alpha P+1}\hat{X}^{\#}(\mathbf{y}+\alpha\mathbf{y}_1+\alpha\mathbf{y}_2+....+\alpha\mathbf{y}_P) \tag{22}$$

where $\mathbf{y}_i$ is a vector of the outputs of the i-th fuzzy model (formed on a basis of D$_i$) where the corresponding coordinate of this vector the output obtained for the corresponding input, that is

$$\mathbf{y}_i = \begin{bmatrix} FM[i](\mathbf{x}_1) \\ FM[i](\mathbf{x}_2) \\ \\ FM[i](\mathbf{x}_N) \end{bmatrix}$$

where $\hat{X}^{\#}$ is a pseudoinverse of the data matrix.

An overall balance captured by Equation (19) is achieved for a certain value of α. An evident tendency of increased impact becomes clearly visible in the sense that higher values of α stress higher relevance of other models and their more profound impact on the constructed model. First, the model is constructed on the basis of D. Second, the consistency is expressed on a basis of differences between the constructed model and those models coming from D$_i$s where the differences are assessed with the use of data D. There is another interesting view of the format of this performance index under minimization. The second component in V plays a role that is similar to a *regularization* term being typically used in estimation problems. However its origin here has a substantially different format from the one encountered in the literature. Here, we consider other data (and models) rather than focusing on the complexity of the model expressed in terms of its parameters to evaluate the performance of the model.

While the semantics of the above performance index in Equation (19) is straightforward, a choice of the value of α requires some attention. To optimize the level of contribution coming from the data sets, we may adhere to the following evaluation process which invokes two fundamental components. As usual, the quality of the optimal model is evaluated with respect to data D. The same optimized model (viz. its prototypes and the parameters of the local regression models) are made available at D$_i$ and the quality of the model is evaluated there with the use of the local data present there. We combine the results (viz. the corresponding squared errors) by adding their normalized values. Given these motivating notes, an index quantifying a global behavior of the optimal model arises in the following form

$$VV = \frac{1}{N}\sum_{\substack{X_K \in D \\ y_k \in D}} (FM(\mathbf{x}_k) - y_k)^2 + \sum_{j=1}^{P} \frac{1}{N_j} \sum_{\substack{x_k \in D_j \\ y_k \in D_j}} (FM(\mathbf{x}_k) - y_k)^2 \tag{23}$$

A schematic view of computing and communication of findings being realized with the aid of Equation (23) is illustrated in Figure 7.

Note that when the fuzzy model FM(.) is transferred to D$_j$, as before we communicate the prototypes obtained at D and the coefficients of the local linear models of the conclusion part of the rules. Likewise as shown in Equation (18), the output of the fuzzy model obtained for $\mathbf{x}_k \in$D$_j$ involves the induced value of membership degree w$_j(\mathbf{x}_k)$ and an aggregation of the local regression models.

Apparently the expression of VV is a function of α and the optimized level of consistency is such for which VV attains its minimal value, namely

$$\alpha_{opt} = arg\ Min\ VV(\alpha) \tag{24}$$

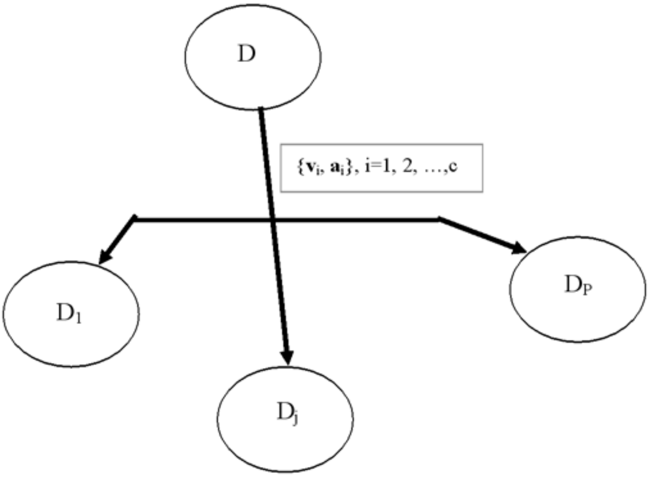


Fig. 7 A quantification of the global behavior of the consistency – based fuzzy model

The optimization scheme in Equation (19) along with its evaluation mechanisms governed by Equation (24) can be generalized by admitting various levels of impact each data D$_i$ might have in the process of achieving consistency. To do so, we introduce some positive weights w$_i$, i=1, 3, …p which are afterwards used in the performance index

$$V = \sum_{\substack{x_k \in D \\ y_k \in D}} (FM(\mathbf{x}_k) - y_k)^2 + \alpha \sum_{j=1}^{P} w_j \sum_{\substack{x_k \in D \\ y_k \in D}} (FM(\mathbf{x}_k) - y_k)^2 \tag{25}$$

Lower values of w$_i$ indicate lower influence of the model formed on a basis of data D$_i$ when constructing the model for data D. The role of such weights is particularly apparent when dealing with data D$_i$ which are in some temporal or spatial relationships with respect to D. In these circumstances, the values of the weights are reflective of how far (in terms of time or distance) the sources of the individual data are from D. For instance, if D$_j$ denotes a collection of data gathered some time ago in comparison to the currently collected data D$_i$, then it is intuitively clear that the weight w$_j$ is lower than w$_i$.

As an auxiliary performance index that expresses a quality of the model for which Equation (19) has been minimized with α being selected with regard to Equation (25), we consider the following expression

$$Q^{\sim} = \frac{1}{N} \sum_{\substack{x_k \in D \\ y_k \in D}} (FM(\mathbf{x}_k) - y_k)^2 \tag{26}$$

The values of Q$^{\sim}$ considered vis-à-vis the results expressed by Equation (26) are helpful in assessing an extent the fuzzy model optimized with regard to data D while achieving consistency with $D_1, D_2, \ldots, D_p$ deteriorates when applied to D over the optimal model being optimized exclusively on a basis of D.

In what follows, we also introduce a computationally effective measure articulating a level of experience consistency obtained for D in the form of *granular* characterization of the parameters of local regression models. Before moving with the details, we elaborate on a way in which individual rules existing in the models formed for D and the data sites $D_1, D_2, \ldots, D_p$ are "synchronized" (aligned).

6.2 The Alignment of Information Granules

The rules forming each fuzzy model have been formed independently at each data site. If we intend to evaluate a level of consistency of the rules at D vis-à-vis the modeling evidence available at D_j, some alignment of the rules becomes essential. Such an alignment concerns a way of lining up the prototypes forming the condition part of the rules. We consider the models obtained at D and D_j, j=1, 2, ..., P with their prototypes $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c$ and $\mathbf{v}_1$[j], $\mathbf{v}_2$[j],..., $\mathbf{v}_c$[j]. We say that the rule "i" at D and the rule "*l*" at D_j are aligned if the prototypes $\mathbf{v}_k$ and $\mathbf{v}_l$[j] are the closest within the collections of the prototypes produced for D and D_j. The alignment process is realized by successively finding the pairs of the prototypes being characterized by the lowest mutual distance. Overall, the alignment process can be described in the following manner:

Form two sets of integers (indexes) **I** and **J**, where **I** = **J** = $\{1, 2, \ldots, c\}$. Start with an empty list of alignments, L= ∅.

```
*Repeat*
    Find a pair of indexes $i_0$ and $j_0$ for which the distance attains minimum
        $(i_0, j_0) = \arg\min_{i,l} \quad ||\mathbf{v}_i - \mathbf{v}_l(j)||$
    The pair $(i_0, j_0)$ is added to the list of alignments, $L = L \cup (i_0, j_0)$
    Reduce the set of indexes I and J by removing the elements that were placed
    on the list of alignments, $\mathbf{I} = \mathbf{I} \setminus \{i_0\}$ and $\mathbf{J} = \mathbf{J} \setminus \{j_0\}$
*until* $I = \emptyset$
```

Once the above loop has been completed, we end up with the list of alignment of the prototypes in the form of pairs (i_1, j_1), (i_2, j_2),..., (i_c, j_c)

6.3 Characterization of Experience-Consistent Models through its Granular Parameters

Once the mechanism of experience consistency has been completed and the local models have been aligned (following the scheme provided in the previous section), we can now look at the characterization of the set of related parameters of the local regression models. In essence, through the alignment of the prototypes at D and D_j, we obtain the corresponding vectors of the parameters of the regression models of the conclusion parts. Denote these vectors corresponding to a certain rule by **a**, $\mathbf{a}_i$, $\mathbf{a}_k$, ..., and $\mathbf{a}_l$ altogether arriving at P+1 of them. If we now consider the j-th coordinate of all of them, we obtain the numeric values a_j, a_{ij}, ..., a_{lj}. The essence of their aggregation concerns their global representation completed in the form of a single fuzzy set. Its modal value is just a_j while the membership function is reflective of the numeric values of the corresponding parameters of the local models. Again for each set of the values of the parameters of the models we apply the principle of justifiable granularity. We consider that a modal value of A_j, that is a_j is given. Let us look at the values of a_{ji} that are lower than a_j, $a_{ji} < a_j$. Denote this set by $\Omega_{.}$, $\Omega_{.}=\{a_{ji}| \, a_{ji} < a_j\}$. We use them to estimate the parameters of the left-hand side of the membership function. The determination of the right-hand side of the membership function is realized in an analogous manner by considering the set Ω_{+} where $\Omega_{.}=\{a_{ji}| \, a_{ji} > a_j\}$ around the modal value of the membership function.

In particular, we can consider a linear form of the membership function. The result of the use of the principle becomes a triangular fuzzy number of the j-th parameter of the local regression model. Denote it by A_j =(a_{j-}, a_j, a_{j+}) with the three parameters denoting the lower, modal, and upper bound of the fuzzy number. Applying the same procedure to all remaining parameters of the vector **a**, we produce the corresponding fuzzy numbers A_1, A_2, ..., A_{j-1}, A_{j+1}, ..., A_n, and A_0. Given them the rule in D reflects the nature of the incorporated evidence offered by the remaining models D_1, D_2, etc. If there is a fairly high level of consistency, this effect is manifested through a fairly "concentrated" fuzzy number. Increasing inconsistency results in a broader, less specific fuzzy number of the parameters. In summary, a certain fuzzy rule assumes the following format

$$\text{If } x \text{ is } B \text{ then } Y = A_0 \oplus A_1 \otimes x_1 \oplus A_2 \otimes x_2 \oplus \ldots \oplus A_n \otimes x_n \tag{27}$$

The symbols $\oplus$ and $\otimes$ being used above underline the nonnumeric nature of the arguments standing in the model over which the multiplication and addition are carried out. For given numeric inputs $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ the resulting output Y of this local regression model is again a triangular fuzzy number Y = <w, y, z> where their parameters are computed as follows

Modal value $y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n$
Lower bound $w = a_0 + min(a_{1.}x_1, a_{1+}x_1) + min(a_{2.}x_2, a_{2+}x_2) + \ldots + min(a_{n.}x_n, a_{n+}x_n)$
Upper bound $z = a_0 + max(a_{1.}x_1, a_{1+}x_1) + max(a_{2.}x_2, a_{2+}x_2) + \ldots + max(a_{n.}x_n, a_{n+}x_n)$

The above process is of the formation of the fuzzy numbers of the local regression model of the rule is repeated for all rules. At the end we arrive at the rules of the form

$$\begin{gathered}
\text{If } x \text{ is } B_1 \text{ then } Y = A_{10} \oplus A_{11} \otimes x_1 \oplus A_{12} \otimes x_2 \oplus \ldots \oplus A_{1n} \otimes x_n \\
\text{If } x \text{ is } B_2 \text{ then } Y = A_{20} \oplus A_{21} \otimes x_1 \oplus A_{22} \otimes x_2 \oplus \ldots \oplus A_{2n} \otimes x_n \\
\ldots \\
\text{If } x \text{ is } B_c \text{ then } Y = A_{c0} \oplus A_{c1} \otimes x_1 \oplus A_{c2} \otimes x_2 \oplus \ldots \oplus A_{cn} \otimes x_n
\end{gathered} \tag{28}$$

Given this structure, the input vector **x** implies the output fuzzy set with the following membership function

$$Y = \sum_{i=1}^{c} w_i(\mathbf{x}) \otimes [A_{i0} \oplus (A_{i1} \otimes x_1) \oplus (A_{i2} \otimes x_2) \oplus \ldots \oplus (A_{in} \otimes x_n)] \tag{29}$$

Owing to the fact of having fuzzy sets of the parameters of the regression model in the conclusion part of the rules, Y becomes a fuzzy number rather than a single numeric value.

7 Experience-Consistent Design of Radial-Basis Function Neural Networks

Radial basis function (RBF) neural networks consist of three layers of processing elements. The first one is made up of source nodes (sensory units). The second layer comprises a collection of highly-dimensional receptive fields (radial basis functions). The output layer consists of a linear unit which linearly aggregates activation levels of the receptive fields. Schematically, the overall structure of the network is presented in Figure 8. The activation level of the i-th receptive field R$_i$ caused by **x** is governed by the expression [7]

$$R_i(\mathbf{x}) = \frac{1}{\sum\limits_{j=1}^{c} \left(\frac{||\mathbf{x}-\mathbf{v}_i||}{||\mathbf{x}-\mathbf{v}_j||} \right)^{2/(m-1)}} \tag{30}$$

where **x** is the input to the network while $\mathbf{v}_i$ is the prototype (center) of the i-th receptive field. The above expression stems from the fact that the receptive fields are formed through fuzzy clustering, say the FCM method and Equation (30) is reflective of the way in which such receptive fields (clusters) have been formed.

The output of the network is formed as a weighted sum of the activation levels of the receptive fields, see Figure 8.

$$\hat{y}_k = w_0 + \sum_{i=0}^{C} R_i(x_k) w_i \tag{31}$$

where w$_i$ is the i-th weight (connection) of the linear neuron.

The determination of the receptive fields, and the prototypes, in particular is realized through fuzzy clustering. The determination of the weights of the output neuron is realized by the minimization of the standard least-square error. Given the linear nature of the problem with respect to, the connections to be determined, the obtained solution leads to the global minimum of the performance index.

For the “P” data sets D[1], D[2], ..., D[P] we construct the corresponding RBF NNs. Following the main design process outlined above, for the ii-th data set we form a collection of receptive fields {R$_i$[ii]}, i=1, 2,...,c which are fully

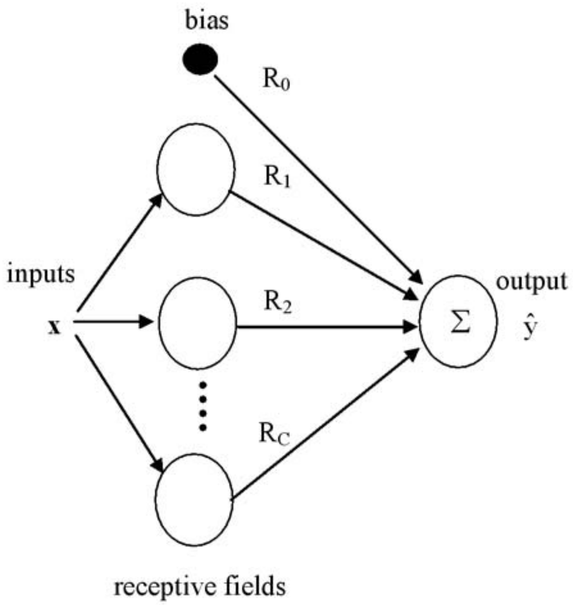


Fig. 8 A schematic view of a RBF neural network (the number of receptive fields is equal to c)

characterized by the prototypes of the clusters $\{\mathbf{v}_i[\text{ii}]\}$. Furthermore the connections of the linear neuron are optimized resulting in the vector **w**[ii]. In summary, the RBF NN is fully characterized by the collection of the prototypes and the vector of the connections which jointly could be described as knowledge acquired from the data D[ii]. To underline this fact, we use the notation **K**[ii] = $\{\ \{\mathbf{v}_i[\text{ii}]\}$, **w**[ii], c$\}$. Note that communicating knowledge is to make **K**[ii] available to the user. The number of clusters (receptive fields) at D and D[ii] is the same. However this assumption is not critical at all and we could envision working with the networks built at different level of detail (different number of the clusters)

In this setting, the idea of the experience-consistent learning of the network translates into an effective usage of experience **K**[1] , **K**[2] ,..., **K**[P] to construct a RBF NN. We further refine this general statement into a functionally meaningful optimization problem. The underlying optimization criterion – performance index comes in the following form

$$\begin{aligned} V = & \sum_{\substack{k=1 \\ x_k, target_k \in D}}^{N} \Big(\sum_{i=0}^{C} w_i R_i(x_k) - target_k \Big)^2 + \\ & + \alpha \sum_{ii=1}^{P} \sum_{\substack{k=1 \\ x_k \in D}}^{N} \Big(\sum_{i=0}^{C} w_i R_i(x_k) - \sum_{i=0}^{C} w_i[ii] R_i[ii](x_k) \Big)^2 \end{aligned} \tag{32}$$

The prototypes of the receptive fields at D[ii] after being used in the context of D give rise to the receptive fields denoted here by R$_i$[ii]. The data set D is composed of input-output pairs { ($\mathbf{x}_k$, target$_k$)}, k=1,2, ..., N. The objective is to minimize V by adjusting the weights of the linear neuron **w**

Given the additive format of Equation (32) which consists of two main components, when minimizing V we attempt to achieve a balance between the model built only on the basis of data D (the first part of Equation (32) and the results produced formerly by the models for data D[ii], ii = 1, 2, ..., P (the second term of Equation (32)). The balance is established by choosing a certain positive value of α. Notably, the higher the value of α, the stronger the impact coming from the experience accumulated in the form of the previously constructed models. If α tends to zero, then the RBF NN is constructed on the basis of the currently available data D. Considering the nature of the second term in the performance index in Equation (3), we could say that it plays a role similar to the regularization mechanism quite often considered in the training of neural networks.

As becomes clear, the result of the learning depends upon the level of impact of the experience-based component (already designed neural networks). While the general tendency could be easily controlled by changing the value of α, choosing its suitable value is not clear at all. An approach we can take comes with the following motivation: the optimal RBF NN should perform well not only on D but also on all other data sets D[1], D[2], ..., D[P]. In other words, we compute a quality of the constructed model on D and then transfer the knowledge **K** to D[1], D[2],..., D[P] and assess the quality of the network over there. The overall performance of the experience-consistent RBF NN is quantified in the form of the following index

$$
\begin{aligned}
G = \frac{1}{N} \sum_{x_k, target_k \in D} \left(\sum_{i=0}^{C} w_i(opt) R_i(x_k) - target_k \right)^2 + \\
+ \frac{1}{N_1} \sum_{x_k, target_k \in D[1]} \left(\left(\sum_{i=0}^{C} w_i(opt) R_i(x_k) \right) - target_k \right)^2 \\
+ \ldots\ldots + \frac{1}{N_P} \sum_{x_k, target_k \in D[P]} \left(\left(\sum_{i=0}^{C} w_i(opt) R_i(x_k) \right) - target_k \right)^2 .
\end{aligned}
\tag{33}
$$

In other words, G expressed by Equation (33) measures the global performance of the optimal neural network when all data are taken into consideration. Apparently G is a function of α and the optimized level of consistency is that for which G attains its minimal value, namely α_{opt} = arg Min G(α).

The optimization scheme Equation (32) along with its evaluation mechanisms governed by Equation (33) can be generalized by admitting the various levels of impact that each data D[ii] could exhibit in the process of reaching consistency. We introduce some positive weights W$_{ii}$, ii=1, 2, ...P which are included in the performance index

$$V = \sum_{\substack{k=1 \\ x_k, target_k \in D}}^{N} \left(\sum_{i=0}^{C} w_i R_i(x_k) - target_k \right)^2 + \\ + \alpha \sum_{ii=1}^{P} \sum_{\substack{k=1 \\ x_k \in D}}^{N} W_{ii} \left(\sum_{i=0}^{C} w_i R_i(x_k) - \sum_{i=0}^{C} w_i[ii] R_i[ii](x_k) \right)^2 \tag{34}$$

Lower values of W_{ii} indicate lower influence of the model formed on a basis of data D_{ii} when constructing the model for data D. The role of such weights is particularly apparent when dealing with data D_{ii}, which are in some temporal or spatial relationships with respect to D. In these circumstances, the values of the weights are reflective of how far (in terms of time or distance) the sources of the individual data are from D. For instance, if D_{jj} denotes a collection of data gathered some time ago in comparison to the currently collected data D_{ii}, then it is intuitively clear that the corresponding value of weight W_{jj} should assume lower values than W_{ii}.

8 Conclusions

We have stressed that the distributed and collaborative nature of systems has to be addressed when dealing with fuzzy models (whose design methodology has been predominantly focused on a centralized development scheme). The collaborative construction of fuzzy rule-based models relies on fuzzy clusters and the schemes of collaborative clustering are of genuine interest with this regard. In the study, we have elaborated on the two main avenues of the formation of the clusters, viz. collaborative clustering and a buildup of hierarchies of clusters which offer some general directions of more detailed algorithmic pursuits.

Acknowledgements. Support from the Natural Sciences and Engineering Research Council of Canada (NSERC) and Canada Research Chair (CRC) is gratefully acknowledged.

References

1. Bezdek, J.C.: Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York (1981)
2. Hoppner, F., et al.: Fuzzy cluster analysis. J. Wiley, Chichester (1999)
3. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall, Englewood Cliffs (1988)
4. Jain, A., Duin, R., Mao, J.: Statistical pattern recognition: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 4–37 (2000)
5. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification 2, 193–218 (1985)
6. Loia, V., Pedrycz, W., Senatore, S.: P-FCM: a proximity-based fuzzy clustering for user-centered web applications. Int. J. of Approximate Reasoning 34, 121–144 (2003)
7. Pedrycz, W.: Conditional fuzzy clustering in the design of radial basis function neural networks. IEEE Transactions on Neural Networks 9, 601–612 (1998)
8. Pedrycz, W.: Collaborative fuzzy clustering. Pattern Recognition Letters 23, 675–686 (2002)

9. Pedrycz, W.: Knowledge-based clustering. J. Wiley, Hoboken (2005)
10. Pedrycz, W., Gomide, F.: Fuzzy Systems Engineering. J. Wiley, Hoboken (2007)
11. Pedrycz, W., Rai, P.: Collaborative clustering with the use of Fuzzy C-Means and Its Quantification. Fuzzy Sets & Systems (2008) (to appear)
12. Strehl, A., Ghosh, J.: Cluster ensembles: a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research 3, 583–617 (2002)
13. Wiswedel, B., Berthold, M.R.: Fuzzy clustering in parallel universes. J. of Approximate Reasoning 45, 439–454 (2007)
14. Zadeh, L.A.: Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. Fuzzy Sets and Systems 90, 111–117 (1997)
15. Zadeh, L.A.: Toward a generalized theory of uncertainty (GTU)—-an outline. Information Sciences 172, 1–40 (2005)

Index

accuracy-complexity tradeoff analysis 131–133, 135, 136, 141, 142, 147–150, 152, 153, 164, 165
adaptation 674, 684, 690, 691, 694
adaptive immune system 531, 533, 535, 537
adaptive systems 33–35, 37
adaptive thresholding 651
agent 362
  competitive, 362
  cooperative, 362
  self-interested, 362
agent-based computational economics 233, 254
agent based system 276
agent behavior estimation 388
  machine learning (ML)-based, 388
  quadratic regression-based, 389
agent negotiation 380
AladdinRescue 345
algorithmic chaining 15
alternating offers protocol 382
ANARCHY 277
anomaly detection 528, 529, 536
ant clustering 640, 641, 643–646, 669
ant colony optimization 113, 115
ant systems 7
architecture
  autonomous agent, 277
  federated, 277
  federated multi-agent, 277
  hierarchical, 277
AROC 411, 428–433
artificial immune systems 7, 18, 527, 555
artificial intelligence 24, 31, 35, 36, 38
artificial life 7, 38
asthma 684
automata 24, 30, 32
  self-reproducing, 23
automatic program generation 34
autonomous-agent designs 244
autonomous systems 530
autopoietic 298

back-marking 207, 214
background subtraction 446
backjumping 204, 206–208, 213, 214, 216
bagging 407, 408, 411, 412, 420
bargaining strategies 257
Bayesian learning 12, 179, 235, 245, 247, 321, 356
Bayesian reinforcement learning 324, 326
Bayesian traders 247–249
beam search 627
Bellman equations 326, 327, 334, 336, 340
binning method 632
bin packing 179, 191
  online, 182, 183, 187, 189, 193
biochemical parameters 677, 687
  blood phosphorus level, 681, 683, 685, 687
  haemoglobin, 677, 687

blackboard
  database system, 12, 276, 277
boids 645
Boolean satisfiability 178, 182–184, 194
bootstrap (ping) 410, 420
Brownian motion 507, 508

Canny edge detector 452, 453
case-based reasoning 12, 20, 179
case based reasoning 673, 676, 680, 689, 694
  diagogue-oriented, 675
  in medicine, 674
chartist 236–240, 246, 254
chromosome 7
classifier ensembles 16, 403, 434
classifiers 446
classifier systems 179, 243
class switching 408, 411, 412, 420, 421
clonal selection 528, 533, 537, 538, 546
clustering 285, 306, 311, 312, 639, 640, 642–646, 652, 669
  fuzzy, 162
  multi-objective, 162, 165
cluster validity measures 613
  partition coefficient, 614
  partition entropy, 614
Cog 23, 25, 38, 39
cognitive immune system 542, 544
collaborative clustering 12, 700, 702–704, 707, 708, 722
collaborative sensor networks 17
collaborative signal processing 475–498
competition 234, 243, 246, 255, 257, 258, 261
competitive agent 362
competitive negotiation 380, 384
complexity
  cognitative, 303, 305
  cognitive, 278, 290–292, 297, 312, 316
  collaboration, 275
  estimator and observer (CEO), 277
  evaluator and observer (CEO), 300
  problem, 275, 276, 278, 287, 291
  process, 278, 286, 310, 315
  real, 277, 286, 287, 302, 312, 313, 315
  self, 300, 301, 313
complexity bounds 284, 295, 300, 315
Complexity Oriented Problem Evaluator (COPE) 304, 316
computational complexity 31
computer vision 16, 441, 466
computer vision techniques 12
concurrent parametric design 281
concurrent product design 282
consistency-based optimization 713
constraint satisfaction 15, 203, 230
cooperative agent 362
cooperative multi-agent system (MAS) 362
cooperative negotiation 380, 384
coordination 321, 322, 324, 328–330, 339, 356, 357
crisp logic 9
crisp logic operators 11
cross-validation 410, 411
crossover
  blend (BLX), 61
crowd dynamics 17, 441, 466
cultural algorithms 113, 114
curse of dimensionality 407, 409
Cyc 23, 25, 38, 39
  OpenCyc, 38
CycL 38
Cycorp 39
cytokines 534, 538, 543

Darwin’s continent-island cycle 561, 562, 570, 571, 577, 579, 580, 592
data mining 11, 625, 626, 631, 634
  multiobjective, 161
data sites 700
data visualization 640, 645, 646, 669
decentralisation 325
decision support system 276
defuzzification 10, 88, 91, 94, 97, 104, 111
dendritic cells 534–537, 548, 550, 551
design structure matrix (DSM) 280, 281, 283, 285–288, 290, 306, 310, 312
diabetes 677
dialysis 673, 677, 678, 680, 682, 684, 686, 694

DIDE 277, 296
differential evolution 113, 116
  fuzzy, 116
Dirichlet distribution 345
disaster response 323, 324, 326, 356, 357
distance measure 614
distributed Karhunen-Loève transform (DKLT) 475, 479–481, 489, 491, 495, 498
distributed systems 475–498
divide-and-conquer 626
DNA replication 623
dodgem cars 507
dual concern model 384
dynamic programming 334
dynamism 325, 340

eCommerce 362
edge detection 446
electric power dispatch 13, 47, 80
equilibrium
  Nash, 381
  negotiation, 381
evolutionary algorithms 6, 7, 14, 627
evolutionary artificial neural networks 261
evolutionary computing 7, 13
evolutionary ensemble design 163
evolutionary fuzzy inference systems 261
evolutionary multi-objective data mining 161
evolutionary multi-objective feature selection 162
evolutionary multi-objective optimization (EMO) 136, 137, 139, 141, 142, 144–146, 154, 161, 163–165
evolutionary optimization 697
evolutionary programming 6, 114, 115
evolutionary support vector machines 261, 263
evolution strategies 6, 83, 91, 110, 114
evolvable hardware 7
experience-consistent schemes 697
experimental economics 234, 246, 254, 255, 257, 260, 267
expert system 276
extended dual concern model 384

face classification 403, 434
face detection 447
face recognition 403, 434
FClust 640, 647–655, 657, 658, 662–665, 667–669
FClust-annealing 651, 655, 661–665, 667–669
FCM algorithm 703
financial data mining 15, 233, 234
finite state machine FSM 323, 331, 332, 337–342, 357
fitness 677, 684
fitness landscape 282
fitness program 677, 678, 684, 687, 693
flexible team formation 366, 371
flock 640–642, 645, 646, 649, 667, 669
flock-based collaborative clustering 639, 670
forward checking 204, 206–208, 214
Fourier analysis 446
Fractal analysis 446
function optimization 588
fundamentalist 237–239, 254
fuzzification 87, 88
fuzzy adaptive genetic algorithms 113
fuzzy ant systems 115
fuzzy C-means (FCM) 700
fuzzy cluster analysis 612
  fuzzifier, 613
fuzzy clustering 19, 700
fuzzy controller 10
fuzzy cultural algorithms 114
fuzzy evolutionary algorithms 14, 83, 119
fuzzy logic 9, 13, 41, 83–85, 89–91, 99, 105, 113, 115, 118, 119, 245, 253, 254, 264, 265
fuzzy modeling 699
  experience-based, 711
  experience-consistent, 700, 711
fuzzy models 699, 722
  rule-based, 699, 700
fuzzy reduction genetic algorithms 113

fuzzy rule 373
fuzzy rule-based systems 83–85, 87–95, 100, 101, 103–105, 131–136, 140–142, 147, 154, 163–165
fuzzy rule set 11
fuzzy sets 9, 10, 20, 697, 698, 700
  type-1, 708
  type-2, 707, 708
fuzzy systems 7, 9, 11, 14

Gaussian distribution laws 502
GDDI cycle 279, 283
General Problem Solver (GPS) 36
genetic algorithms 6, 25, 83, 99, 100, 104, 112, 114, 132, 138, 154, 165, 179, 194, 195, 242, 243, 251–253, 255, 256, 261–263, 567, 568, 571, 576, 581, 582, 588
  Breeder Genetic Algorithm (BGA), 561, 588, 590, 592
    line recombination, 588, 589
    mutation, 588, 589
  building blocks, 582
  centralized, 562
  co-evolutionary, 562
  MPX crossover, 582
  multi-objective, 131, 140
  one-island model, 571
  parallel, 562, 568
  Parallel Genetic Algorithm (PGA), 561, 562, 568, 571, 573, 581, 583, 588
  schema theorem, 582
  the island model, 571
  the isolating by distance model, 571
  the stepping-stone model, 561, 571
genetic fuzzy classifier systems 253
genetic fuzzy systems 14, 83, 119, 132–134, 136, 141, 165
  multi-objective, 14, 131, 136, 140–142, 161, 162, 164, 165
genetic programming 6, 14, 83, 91, 99, 100, 113, 114, 177, 198, 235, 243, 244, 250, 255–257, 260, 262, 265, 266
  fuzzy, 114
  multi-population genetic programming (MGP), 256
Gibbs-Boltzmann distribution 239
global economy 698
GO 629, 632
Google 24
granular computing 697
graph partitioning problem 584, 585, 587, 588

haemodialysis 677
Hamming distance 706
HEARSAYII 304
hematocrit 688
heterogeneous agents 234, 235, 243, 254, 260, 267
hierarchical clusters of clusters 710
hierarchies of clusters 20, 700, 722
hill climbing 627
homeostasis 531
human-centric systems 698
hyper-heuristics 12, 14, 177, 198

ILOG Solver 203
IMMUNE 275
immune networks 528
immuno-engineering 18, 553
information granules 20, 698–700
inheritance system
  behavioral, 565
  epigenetic, 565
  genetic, 565
  symbolic, 565
innate immune system 533–535
integrative systems 312
intelligence
  collaborative, 23, 24
  collective, 23
  individual, 23
intensity-dependent noise 606
intercontact time (between transmitting agents) 503, 507, 511, 513, 514, 516–518, 522
intergrative method 624
intracontact time (between transmitting agents) 516
iterated prisoner's dilemma 561, 569, 572, 573
iTRAQ^{TM} 604

JANUS 23, 25, 40, 41

K-means 642, 643, 645, 646, 652–654, 657, 664, 667, 670
K-means+FCLust hybrid-annealing algorithm 655
K-means+FClust hybrid algorithm 639, 641, 652–655, 657, 664–669
k-nearest-neighbors 235, 247–249
Kalman Filter 357
Karhunen-Loeve transform 418
kin selection 564, 579
knowledge base 678
knowledge based systems 673

Lagrange multipliers 704
learning 23, 24, 27–30, 36, 37, 39–41
learning classifier 544
learning classifier systems 98
learning to learn 195
least-square error 719
Lighthill Report 5
LIKE-PART 633, 634
likelihood curves 19, 610
likelihood ratios 12, 19, 20, 623, 625, 630–632, 634, 635
Lin-Kernighan algorithm 584
  iterated, 583, 588
linear covariance 283
local regression models 713
Logic Theorist (LT) 35, 36
logit model 239, 240, 265
long-term team formation 365, 370

machine intelligence 25, 27, 30, 36, 37, 42
machine learning 39
  genetics-based, 154
  multi-objective, 136, 161
  multi-objective fuzzy genetics-based, 140
machine learning (ML)-based agent behavior estimation 388
macrophages 534, 535
Markov Decision Process MDP 333
Markov decision process MDP 324, 326, 327, 332, 334–336, 339, 341
  belief, 335, 336
  fully observable, 337
  partially observable, 337
  partially observable POMDP, 327, 331, 333–337, 341, 343, 345, 356
mass spectrometry 602
  ESI, 602
  LC-MS, 603
  LC-MS/MS, 603
  MALDI, 602
  MS/MS, 602
MATE 277
MATLAB 422
Means-Ends Analysis 36
membership degree 613
meta-granules 699
metabolic pathways 623
MetaMorph 277, 307
MINOS 40
mirroring 234, 255–257
mirror neurons 24
missing data 677, 678, 686, 687, 693, 694
  restore, 673
model
  dual concern, 384
  extended dual concern, 384
model based reasoning 677
model failure 681
Modern Synthesis 562, 564, 565
Monte Carlo simulation 282, 283, 293, 300, 315
multi-agent systems 11, 12, 15, 16, 276, 277, 289–292, 294, 296, 297, 321, 343, 361
  bandwidth limited, 324
  competitive agent, 16
  cooperative, 362
  cooperative agents, 16
  decentralisation, 324
  dynamism, 324
  openness, 324
  partially-observable system, 324
  self-interested, 362
  self organization, 12
multi-objective evolutionary algorithms 7, 13, 47, 80, 101, 113
multi-objective genetic programming 163
multi-objective optimization 113
  fuzzy based, 49, 50, 60, 62
multi-objective particle swarm optimization 79

Munich Information center for Protein Sequences (MIPS) 625, 629, 632
mutation
  non-uniform, 61

N-type designs 235, 236, 241, 242, 244
Nash equilibrium 381
natural computing 3
natural immune system 527, 555
natural killer cells 533–535
natural selection 562, 563, 565, 567, 569, 581
nearest neighbor 406, 408, 410, 412, 413, 427, 431, 432
nearest neighbour search 674
negative exponential distribution 503, 510, 512, 516
negative selection 528, 533
negotiation
  agent, 380
  competitive, 380, 384
  cooperative, 380, 384
negotiation equilibrium 381
negotiation protocol 381
negotiation strategy 381
neural networks 5, 7–9, 13, 17, 20, 28, 37, 40, 41, 83, 91, 131, 132, 163, 165, 194, 245, 261–264, 407, 410, 412, 418, 420, 478, 479, 483–485, 491, 492, 495, 496, 697, 700
  approximate identity, 478
  backpropagation, 8, 36
  ensemble design, 163
  feedforward, 478
  fuzzy, 262, 264, 265
  genetic fuzzy, 92
  Kohonen nets, 9, 410
  Levenberg-Marquardt, 16, 403, 407, 408, 412, 418
  MATLAB toolbox, 422
  McCulloch-Pitts theory, 31
  multi-layer perceptrons, 478
  multi-objective, 163
  non-dominated, 163
  radial-basis, 719
  radial basis functions, 475, 478, 484
  reinforcement learning, 8
  self organizing networks, 9
  supervised learning, 8
  unsupervised learning, 8
neuro-fuzzy systems 132, 133
Niched Pareto Genetic Algorithm (NPGA) 49, 54, 56, 62, 64, 66, 68–74, 76
Niched Pareto Genetic Algorithm 2 (NPGA 2) 79
No Free Lunch Theorem 204, 220
Non-dominated Sorting Genetic Algorithm (NSGA) 49, 54, 55, 62, 63, 66, 69–76
Non-dominated Sorting Genetic Algorithm II (NSGA-II) 59, 79, 101, 136–140, 146, 147, 154–158, 162, 164
numerical solutions 687
  binary, 687
  estimated, 687
  exact, 687

objective function 613
one-shot team formation 365, 369
Ontix 284
OntoSpace 284
opportunist networks 17, 501, 523
optical flow techniques 443
overfitting 627, 628
overgeneralization effects 404, 405, 427

PANDEMONIUM 40
parallel evolutionary algorithms 7, 18
  fuzzy, 113
parathyroid hormone 681, 683–685, 687, 689
Pareto distribution 510, 512, 516–518
PART 626, 631, 632, 634
partially-observable system 323, 330, 336
particle swarm clustering 640–642, 646
particle swarms 7, 20, 113, 115, 624, 640, 642, 643
  fuzzy, 115
partition matrices 700
  induced, 701
partition matrix 704
  induced, 706

peptide 602
phagocytes 533, 534
physiological parameters 677, 687
burned calories, 677, 687
haemoglobin, 682
lung ventilation, 677, 687
maximal power, 687
maximal power achieved, 677, 682
oxygen pulse, 677, 678, 682, 687
oxygen uptake, 677, 678, 682
performed work, 678
physiotherapy
exercises, 677
plasma volume 688
pocket trace recorder (PTR) 514–518
point matching 451
principle components analysis (PCA) 406, 408–410, 412–418, 422, 427, 430–432
probability density functions 460, 466
problem connectivity 310
PROLOG 627
protein-protein interactions 623–625, 630, 633–635
proteins 602
protein synthesis 623
proteome 602
proteomics 602
protocol
alternating offers, 382
prototype 615
initialisation, 616
update, 616
prototypes 700
proximity matrix 705
PRTools 422
punctuated equilibrium 581

quadratic regression-based agent behavior estimation 389
Quickest First Principle 15, 203, 230

random subspace 408, 411, 412, 420, 421, 433
real-coded genetic algorithm 61
REBA algorithm 15, 205–207, 212–221, 228, 229
receiver operating characteristic (ROC) 625, 626, 632–634
regulation 604
regulation factor, expression ratio 606
reinforcement learning 179, 245, 246, 252
representative agents 234, 235
retrieval 674, 675, 677, 680, 682, 683, 690, 694
retrieval nets 674
RoboCup 362
Robocup Rescue 345
Rosetta Stone Method 624, 629
rough sets 265, 266, 698
rule based reasoning 676
rule discovery algorithm 626
rule induction 12, 19, 20, 623, 625–627, 630–632, 634

scatter search 582
self-interested agent 362
self-interested multi-agent system (MAS) 362
self-organizing map 443, 446, 462
self complexity 284
selfish gene concept 564
sensor networks 476–498
sentential calculus 35
separate-and-conquer 626, 627, 631
sets 698
SHRDLU 5
simulated annealing 7, 179
simulated evolution 36
single classifiers 405, 411, 422, 431, 433
soft computing 3
speckled computing 548
statistical methods 678, 686
clustering, 677
correlation, 678
Fisher's exact test, 679, 682, 693
frequency table, 679
logistic regression, 677
Pearson Chi-square test, 679
statistical model 694
story board 505
Strength Pareto Evolutionary Algorithm (SPEA) 49, 54, 58, 59, 62, 65, 66, 68–76, 78, 79

Strength Pareto Evolutionary Algorithm 2 (SPEA2) 59, 79, 101
Structuration Theory 307, 310
supervised learning 447
Support Vector Machine 624
support vector machine 406, 408, 410, 412, 413, 415, 416, 422, 427, 431–433
swarm intelligence 234, 639, 670

tabu search 179
Teacher system 196
team-formation mechanism 365
team formation
  flexible, 366, 371
  long-term, 365, 370
  one-shot, 365, 369
temporal pyramidal analysis 452
tracking 447, 460
trait impressions 403, 434
travelling salesman problem 582–584
tree-hash retrieval 674
trigger time 508, 511, 512
Turing machine 29, 31, 33, 37
Turing test 26, 39, 42

uncertainty 321, 322, 326, 340, 356
unmanned aerial vehicles UAVs 321, 323

variables
  manufacturing, 281
  process, 281
  shape, 281
  sizing, 281
  topology, 281
Vector Evaluated Genetic Algorithm (VEGA) 53
visualization panel 640, 646–649, 656, 657, 664, 667, 669, 670

WEKA 631
Wikipedia 24
wireless sensor networks 702

zero-intelligence agents 245