Pietari Pulkkinen

**Multiobjective Genetic Fuzzy Systems in Data Classification and Regression**

Julkaisu 950 • Publication 950

Tampere 2011
Pietari Pulkkinen

Multiobjective Genetic Fuzzy Systems in Data Classification and Regression

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Sähkötalo Building, Auditorium S4, at Tampere University of Technology, on the 18th of February 2011, at 12 noon.
Abstract

This thesis presents data-driven methods to learn interpretable and accurate fuzzy models (FMs) for classification and regression problems. When FMs are identified based on data, the main advantage of FMs, namely interpretability can easily be deteriorated unless appropriate learning method is used.

Multiobjective evolutionary algorithms (MOEAs) are selected in this thesis to learn FMs because they are very flexible learning methods and have proved to be robust in many learning tasks. This approach to use MOEAs to learn FMs is often referred to as multiobjective genetic fuzzy systems (MGFSs). In this thesis MGFSs are applied to learn the rules, select input variables, determine the granularity of each input and output variable and to tune the membership function (MF) parameters. The goal is to maximize the accuracy and interpretability, which are conflicting objectives. MGFSs can find a set of Pareto optimal FMs presenting different trade-offs between accuracy and interpretability. After the user has seen the available choices, he/she selects one or more of them based on the preferences.

MGFSs face challenges especially in high-dimensional problems (i.e. the number of input variables is high). That is because the search space is large, which makes it difficult to find good FMs. This thesis proposes initialization methods to remove irrelevant input variables, which reduces the search space and eases the further optimization by MOEA. Another challenge is related to MFs tuning which usually improves accuracy but deteriorates transparency of fuzzy partitions unless adequate tuning strategy is used. Two solutions for that problem are proposed. First, an interpretability index is used to measure the transparency of fuzzy partitions and the purpose is to optimize its value. The second proposal uses dynamic constraints to guarantee that the user specified transparency conditions are met by each FM at any given phase of optimization. This proposal reduces the number of objective functions by one which improves the search efficiency of MOEAs.

Altogether five MGFSs are proposed in this thesis. Three of them are designed for classification problems and two of them for regression problems. They
are evaluated on seven classification and 12 regression problems. Results comparisons show that they outperform several MGFSs in the literature. Finally, through an industrial application, it is shown that MGFSs are suitable for learning FMs to be used as a reasoning mechanism in a bioaerosol detector.
Preface

This thesis work gave me tremendous opportunities to learn and work with interesting ideas, projects, and people. It was made possible especially by the funding from Tampere University of Technology’s graduate school. I am also very grateful for the funding from Emil Aaltosen säätiö, Tekniikan edistämissäätiö, Wärtsilän rahasto, and Automaatiosäätiö.

I want to thank my supervisor, professor Hannu Koivisto, for guidance and for giving me interesting project and teaching responsibilities which enabled me to develop my skills. I especially enjoyed working in a project together with the people from Dekati Ltd. and Finnish defense forces.

I am thankful for the pre-examiners, professors Hisao Ishibuchi and Jarmo Alander, whose comments and critique improved the thesis. Professor Ishibuchi also assigned me to be a reviewer of IEEE Transactions on Fuzzy Systems journal, which taught me the difference between good and not so good articles.

I also had a great opportunity to conduct research overseas at Florida Institute of Technology in professor Eraldo Ribeiro’s Computer Vision and Bio-Inspired Computing research group. This allowed me to learn new ideas, experience different work culture, and meet great people.

Obviously, I had great discussions with many people at the Department of Automation Science and Engineering where I worked most of the time when writing this thesis. Special thanks are due to M.Sc. Tomi Helin and Dr. Mikko Laurikkala.

My wife, Silja, has been a great wife and supported me throughout this process. She has definitely helped me to write better and to prioritize different tasks adequately. Finally, without God, neither me nor this thesis would ever exist.

Tampere, January 2011

Pietari Pulkkinen
Contents

Abstract iii
Preface v
List of Publications xi
Supplementary Publications xiii
List of Symbols xv
List of Acronyms xvii

1 Introduction 1
1.1 Contributions of this Thesis to the MGFS Field . . . . . . . . . . 4
1.2 Contributions of the Publications of this Thesis . . . . . . . . . . 8
1.3 Contributions of this Thesis as a Whole . . . . . . . . . . . . . . 10
1.4 Author’s Contribution . . . . . . . . . . . . . . . . . . . . . . . . 11
1.5 Organization of this Thesis . . . . . . . . . . . . . . . . . . . . . 11

2 Components of Multiobjective Genetic Fuzzy Systems 13
2.1 Fuzzy Models . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
2.1.1 Input and Output Data . . . . . . . . . . . . . . . . . . . . . 15
2.1.2 Gaussian Bell Membership Functions . . . . . . . . . . . 16
2.1.3 Fuzzy Rules . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
2.2 Evolutionary Algorithms . . . . . . . . . . . . . . . . . . . . . . 18
2.3 Multiobjective Evolutionary Algorithms . . . . . . . . . . . . . . 21
2.4 NSGA-II: The Nondominated Sorting Genetic Algorithm II . . . 24
2.5 Multiobjective Genetic Fuzzy Systems . . . . . . . . . . . . . . . 25
2.5.1 Some Common Initialization Methods . . . . . . . . . . . . 26
5.4 Discussion ................................................................. 90

6 Conclusions and Further Research Directions .................. 93

Bibliography ....................................................................... 97

Publications ....................................................................... 107
List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by P1–P5.


Supplementary Publications

The following publications are either extended in the publications P1–P5 or they are slightly out of scope of this thesis.


List of Symbols

The following list includes the symbols which frequently appear in the overview of this thesis. Some of the symbols listed here have different meaning in some of the publications P1-P5.

\( a \)  Gbell membership function parameter defining the width
\( A \)  A Matrix containing the rule antecedents
\( B \)  Input fuzzy set
\( b \)  Gbell membership function parameter defining the shape
\( c \)  Gbell membership function parameter defining the center
\( \bar{C} \)  Center of output membership function
\( D \)  Number of data points
\( F \)  Number of input variables in a fuzzy model
\( G_{\text{tot}} \)  Number of generations
\( M_A \)  Number of active membership functions in input variable
\( M_{\text{DT}} \)  Number of membership functions in input variable of a fuzzy model obtained by transforming a decision tree into a fuzzy model
\( M_{\text{in}} \)  Maximum number of membership functions in each input variable
\( M_{\text{max}} \)  Maximum number of membership functions in each input and output variable
\( M_{\text{out}} \)  Maximum number of membership functions in output variable
\( n \)  Number of input variables in a data set
\( n_s \)  Number of input variables selected from \( n \) by the initialization method
\( N_{\text{pop}} \)  Population size
\( o \)  A vector containing the output membership function parameters
\( P \)  A matrix containing the input membership function parameters
\( R \)  Number of rules
\( R_{\text{cond}} \)  Number of rule conditions (total rule length)
\( s \)  A vector containing the rule consequents
\( U \)  Universe of discourse
\( x_k \)  Data point \( k \)
\( X \)  Input matrix
\( y_k \)  Actual output for data point \( k \)
\( \hat{y}_k \)  Estimated output for data point \( k \)
\( y \)  Output vector
\( \alpha \)  The maximum membership value at the intersection of two membership functions
\( \beta \)  The minimum membership value at any point of universe of discourse
\( \beta_i(x_k) \)  Fulfillment of rule \( i \) for data point \( x_k \)
\( \gamma \)  The maximum membership value at the center of another membership function
\( \mu_B \)  Membership function of fuzzy set \( B \)
\( \delta \)  Total number of membership functions in initial fuzzy model
\( \theta \)  Number of parameters to be optimized
\( \rho \)  Number of parameters used to define a membership function
List of Acronyms

CV    Cross-validation
DT    Decision Tree
EA    Evolutionary Algorithm
EP    Evolutionary Programming
ES    Evolutionary Strategies
FIS   Fuzzy Inference System
FM    Fuzzy Model
FP    False Positive
Gbell Generalized Bell
GA    Genetic Algorithm
GP    Genetic Programming
GFS   Genetic Fuzzy System
MF    Membership Function
MGFS  Multiobjective Genetic Fuzzy System
MOEA Multiobjective Evolutionary Algorithm
MOP   Multiobjective Problem
MSE   Mean Squared Error Divided by Two
NSGA-II Nondominated Sorting Genetic Algorithm II
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAES</td>
<td>Pareto Archived Evolution Strategy</td>
</tr>
<tr>
<td>SPEA2</td>
<td>Strength Pareto Evolutionary Algorithm 2</td>
</tr>
<tr>
<td>TP</td>
<td>True Positive</td>
</tr>
<tr>
<td>UoD</td>
<td>Universe of Discourse</td>
</tr>
<tr>
<td>WM</td>
<td>Wang-Mendel</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Developing models to mimic real-world systems is important in many fields of science and engineering. A model can be used to predict an output of the system based on given input(s), to simulate the system, to analyze the system etc. Models can also be used in control, optimization, fraud detection and fault diagnostics. Numerous models have been proposed and they can be in the broader sense divided into linear and non-linear models and to static and dynamic models. Naturally, the quality of the model determines the quality of the solution to the problem at hand [70]. For example, when the purpose is to predict temperature, a good model gives accurate predictions. On the other hand, if the purpose is to use the model to expose frauds, a good model should tell the user whenever a fraud is taking place but should not give many false alarms, which are often costly in several ways. Moreover, accuracy may not be the only goal but it may be also desired that the model is not a black-box but that its functioning can be understood. In other words, it is desired that the model is interpretable, which is beneficial because it provides insights to the process and makes it possible to verify that the model actually works reasonably. For example, when a model suggests that a fraud is taking place, it is vital to understand why it is doing so. Moreover, update or modification of the model to changing situations is more likely to be successful when the model is interpretable. In fact, models which can not be interpreted may not be applied in practice due to lack of confidence in them [34].

While there are many model types which can accurately model linear and non-
linear processes, there are not so many model types which can also be interpretable. Fuzzy models (FMs) are one of them. Their reasoning is based on "If-then" fuzzy rules, which resemble human reasoning. Therefore they can be interpretable especially when created based on expert knowledge. Often when expert knowledge is not available or when it is too difficult to create an FM based on expert knowledge, data-driven FM identification is used instead. That, however, may lead to low interpretability if the applied learning method only tries to improve the accuracy and neglects the interpretability. In fact, many of the earlier works in data-driven FM identification fall into that category [61, 75, 87, 88].

Interpretability and accuracy are conflicting objectives. When accuracy is improved, interpretability is deteriorated and when interpretability is improved, accuracy is deteriorated. Depending on the user preferences and the problem at hand, different levels of interpretability and accuracy are naturally required. In order to have some certainty that the obtained FM presents appropriate trade-off, given the user preferences and the problem at hand, it is of interest to find a set of trade-off FMs before making the final decision about which FM to use in practice. Evolutionary algorithms (EAs) are often used to identify trade-off FMs by setting weights for the conflicting objectives. As a result of each EA run, a single FM presenting a trade-off between accuracy and interpretability is obtained. Thus, multiple runs with different weights are required to obtain a set of trade-off FMs. Furthermore, it is not certain that with every run a distinct FM is found [30].

This thesis uses multiobjective evolutionary algorithms (MOEAs) to optimize the parameters of FMs, which means that user is not required to set any weights. Moreover, as a result of a single optimization run, a set of trade-off (i.e. Pareto optimal) FMs is obtained. After the optimization run, a suitable FM based on the needs and preferences can be selected. The approaches which use EAs and MOEAs to identify FMs are called genetic fuzzy systems (GFSs) and multiobjective genetic fuzzy (MGFSs), respectively.

The FM identification task is challenging because the number of parameters to be optimized is usually high. In other words the search space is large and it increases when more parts of FMs (e.g. rule base, parameters of membership functions) are optimized and when high-dimensional problems (i.e. problems with many input variables) are considered. The MGFSs in this thesis learn the rules and simultaneously perform membership functions (MFs) tuning. The search space
is reduced by initialization methods, which remove irrelevant input variables and therefore ease the further optimization by MOEA.

Another challenge is related to MFs tuning which usually improves the accuracy of FMs but deteriorates the transparency of fuzzy partitions if appropriate tuning strategy is not used. If the transparency of fuzzy partitions is deteriorated, it is difficult, if not impossible, to give reasonable linguistic meaning to the fuzzy sets. That makes the fuzzy rules and the whole FM less interpretable. An example of a fuzzy partition which is not transparent is shown in Fig. 1.1(a) and an example of a transparent fuzzy partition is given in Fig. 1.1(b). In this thesis global fuzzy partitions are used which means that the same MF definitions are used in all rules. This leads to more transparent fuzzy partitions than the application of local fuzzy partitions in which the MFs can be defined differently in each rule.

In order to avoid fuzzy partitions that are not transparent, two solutions are proposed in this thesis. First, a fuzzy partition transparency index is included as an objective to be optimized. Second, a dynamic MFs tuning strategy, which guarantees that each fuzzy partition in each FM meets certain transparency criteria is proposed.

Altogether five MGFSs are proposed in the publications P1- P5 of this thesis. MGFSs in P1, P2, and P5 are applicable to classification problems and MGFSs in P3 and P4 to regression problems. The MGFSs are tested on a set of 19 data sets. 12 of them present regression problems and the rest seven classification problems. The results are compared to many recent MGFSs in the literature. The proposed MGFSs perform well in comparison to them and especially the MGFSs in P4 and P5 show very good performance.

Finally, the usefulness of the MGFS proposed in P2 is validated in an industrial project. The aim of the project was to develop an accurate and interpretable FM, which can be used to infer based on real-time measurements whether there are harmful or safe bioaerosols in the air. The bioaerosol detector was at a prototype phase during the project. Therefore, when a false alarm was raised, it was important to verify whether it was caused by the FM or the bioaerosol detector itself. Interpretable FM was helpful to track down the root cause of false alarms. As the bioaerosol detector went through certain developments, the data it provided slightly changed. That required fine tuning of the FM, which was possible partly
Figure 1.1: An example of a fuzzy partition for which linguistic values are difficult to define due to lack of transparency (a) and a transparent fuzzy partition for which linguistic values, such as "Very small", "Small", "Medium", "Large", and "Very large", could easily be given (b).

due to its interpretability. As a result of the project, an FM was installed into the bioaerosol detector.

1.1 Contributions of this Thesis to the MGFS Field

The literature considering GFSs and MGFSs is rich and it can be said that the field has reached certain maturity [40]. It is, therefore, necessary to present a brief survey of the field in order to understand the contribution of this thesis. However, this survey is far from exhaustive and the reader is directed to [40, 45] for more information. The survey mainly focuses on MGFSs using second generation MOEAs (e.g. nondominated sorting genetic algorithm II (NSGA-II) [30], strength Pareto evolutionary algorithm 2 (SPEA2) [98], the Pareto archived evolution strategy (PAES) [65]), which are the state-of-the-art MOEAs at the moment. Furthermore,
the survey focuses on MGFSs applicable to classification and/or regression problems using linguistic Mamdani FMs [66] (i.e. the rule consequent is a fuzzy set or a class label). Therefore, approaches which use Takagi-Sugeno (TS) [86] FMs are excluded. That is done because Mamdani FMs are often considered more interpretable than TS FMs, which have linear function as their rule consequent. TS FMs, on the other hand, are often more accurate than Mamdani FMs.

Table 1.1 presents some of the second generation MGFSs for identification of linguistic FMs for classification and regression problems. All publications of this thesis (P1 - P5) are included to the Table so that the reader may have better understanding how their components differ from other MGFSs in the literature. In this Table rule selection means that a rule is selected from a set of candidate rules, whereas rule learning means that appropriate rule conditions are learned by MGFS. Because rules are an essential part of FMs, all but one [15] of the approaches either learn the rules or select them out of candidate rules.

MFs tuning is another component which is often employed by MGFSs in order to improve accuracy. Some of the MFs tuning methods, however, deteriorate the transparency of fuzzy partitions by allowing the MFs to overlap too much and therefore making the assignment of linguistic values difficult. In order to avoid that the approach [19] uses fixed MFs which are evenly distributed and uniformly shaped. Moreover, each input variable is partitioned with the same amount of MFs. That approach obviously leads to very good transparency of fuzzy partitions but usually does not lead to as accurate FMs as when MFs are optimized. The approach [12] goes a step ahead by learning the granularity (i.e. the number of MFs) for each input variable. Another way to improve the accuracy without tuning the MF parameters is to select MFs of fuzzy rules from four different fuzzy partitions (which are not optimized but pre-specified) [47, 50, 51, 69]. Those fuzzy partitions present different granularities and they are usually selected, such that, they are highly transparent. However, when MFs are taken from different fuzzy partitions, the resulting global fuzzy partition is not necessarily transparent [72].

A commonly used method for MFs tuning is the 3-parameter MFs tuning method with static constraints [8, 37], [P1, P3]. Those static constraints remain fixed during the MGFS run and need to be defined before the run which is a difficult task. That is because whenever the MF parameters are altered, the constraints which were initially relevant, may not be relevant anymore but should also be
Table 1.1: Some second generation MGFSs applied to identification of linguistic FMs.

<table>
<thead>
<tr>
<th>Problem type</th>
<th>Year</th>
<th>Reference</th>
<th>Rule selection</th>
<th>Rule learning</th>
<th>MFS tuning</th>
<th>Transparent fuzzy partition</th>
<th>Input variable selection:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>2004-5</td>
<td>[50, 69]</td>
<td>✓</td>
<td></td>
<td></td>
<td>Not always</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2007</td>
<td>[51]</td>
<td>✓</td>
<td></td>
<td></td>
<td>Not always</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2007</td>
<td>[47]</td>
<td>✓</td>
<td></td>
<td></td>
<td>Not always</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2008 P1</td>
<td></td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td>Not always</td>
<td>✓ ✓</td>
</tr>
<tr>
<td></td>
<td>2008 P2</td>
<td></td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2009 P5</td>
<td></td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2010</td>
<td>[33]</td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2007-9</td>
<td>[8, 37]</td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td>Not always</td>
<td>✓ ✓</td>
</tr>
<tr>
<td></td>
<td>2008-9</td>
<td>[7, 32]</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2009</td>
<td>[12]</td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2009 P3</td>
<td></td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td>Not always</td>
<td>✓ ✓</td>
</tr>
<tr>
<td></td>
<td>2009 [17]</td>
<td></td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2009 [15]</td>
<td></td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2010 P4</td>
<td></td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2010 P5</td>
<td></td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2010 [39]</td>
<td></td>
<td>✓ ✓</td>
<td></td>
<td></td>
<td>✓ ✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

updated to correspond the current MF parameters. Consider a simple example with two MFs A and B. Let MF A be located on the left hand side of MF B. In that case MF A should be allowed to move towards right a certain amount so that it will not overlap too much with MF B. However, if MF B also moves towards right, MF A on its left hand side could actually move more towards right without overlapping too much with MF B. However, because the constraints are static, this is not possible.

Linguistic 2-tuple MFs tuning strategy [5] also relies on static constraints. Therefore, the limitations mentioned in the previous paragraph apply to this tuning strategy as well. However, since MFs are defined by only one parameter, definition of static constraints is rather simple. This tuning strategy requires that all MFs are uniformly shaped and only the lateral displacements of MFs are modified. This
reduces the search space and therefore some of the recent MGFSs employ this tuning strategy \cite{7,32}. However, because the shapes of the MFs are fixed, this tuning strategy may not lead to as good accuracy as when the shapes are also subject to optimization.

In order to use 3-parameter MFs tuning to obtain higher level of accuracy without deteriorating the transparency of fuzzy partitions, several transparency indexes (or penalty factors) have been proposed. Their purpose is to measure the quality of fuzzy partitions and they are subject to optimization. P2 was one of the first publications in the literature which applied a second generation MOEA to concurrently learn rules and to tune MFs with 3-parameter MFs tuning while preserving the transparency of fuzzy partitions by minimizing a penalty factor. After that several other works have followed \cite{13,15,39}.

P4 and P5 introduced 3-parameter dynamic MFs tuning strategy which overcame the aforementioned limitations of MFs tuning with static constraints. In this tuning scheme, constraints are not fixed but updated every time an MF parameter is modified. Therefore, this tuning scheme can lead to highly accurate FMs without deteriorating transparency of fuzzy partitions. Moreover, user can set the transparency level he or she desires and all FMs will always meet those demands. Furthermore, since transparency of fuzzy partitions is maintained by constraints, there is no additional objective function related to transparency of fuzzy partitions. This improves the search ability of current MOEAs which do not perform well when the number of objectives functions is high \cite{53}.

The last two columns of Table 1.1 indicate whether input variables are reduced during the initialization phase and/or during the learning phase (i.e. MOEA optimization). When input variables are already reduced in initialization phase, the search space is naturally reduced and the MOEA optimization is made easier. It can be seen from Table 1.1 that only the publications of this thesis remove input variables in initialization phase.

In P1 and P2 that is done by decision tree (DT) initialization proposed in \cite{3}. In P3 this approach, which was originally suitable for classification problems only, was modified to suit regression problems as well. Finally, another initialization method, which combines the benefits of Wang and Mendel (WM) initialization and DT initialization was proposed in P4. This initialization method results into
more compact initial FMs than when WM method is used. Moreover, the fuzzy partitions are transparent, which is not always the case when DT initialization is used. Since in this thesis, regression problems up to 26 input variables and classification problems up to 60 input variables are considered, the role of initialization is rather important. Finally, in all publications of this thesis, the number of input variables can be further reduced during the MOEA optimization.

1.2 Contributions of the Publications of this Thesis

The contributions of the five publications of this thesis are summarized in the following. The publications are listed in the order they were written, that is, P1 and P5 are the first and the last of the publications, respectively. One may, however, notice that P5 was published before P4 but actually P4 was written before P5.


DT initialization method [3] was applied in order to reduce the number of input variables, to obtain initial fuzzy partitions, and to create initial rules. After that, MOEA was used to learn rules, granularity, number of input variables and to tune MFs. To the best knowledge of the author, this is the first publication which combines DT initialization and MOEA and the first publication which simultaneously performs rule learning, granularity selection, input variable selection, and MFs tuning with MOEA. This claim is also supported by the following quotation by Alcalá et al. [7] in which references [21] and [22] refer to P1 and P2, respectively:

None of these approaches, however, consider the learning or tuning of membership function (MF) parameters, which involves a more complex search space. It is only recently that two similar MOEAs have been proposed to perform a tuning of the MFs while evolving the antecedents of a previously obtained rule base (RB) in classification problems [21], [22].

In P1 transparency of fuzzy partitions was neglected. Here, however, it is taken into account by optimizing a fuzzy partition transparency index as one of the fitness functions. This transparency index is a further development of the transparency index proposed in [62] and it reduces the effects of relaxed covering [89]. The proposed MGFS is applied to identify FMs for a bioaerosol detector, which is a novel application area. As this was a confidential project, P2 can only reveal fairly general details about the signals and the functioning of the bioaerosol detector.


Unlike P1 and P2 which deal with classification problems, this publication deals with regression problems. The main contribution is the modification of the DT initialization method [3] for regression problems. It is based on discretization of the output data to some number of regions, which can be presented as classes. This converts a regression problem to a classification problem which enables the application of DT initialization [3].


This publication deals with regression problems but unlike P3 which uses 3-parameter MFs tuning strategy with static constraints, here a novel 3-parameter MFs tuning strategy with dynamic constraints is proposed. This improves the transparency of fuzzy partitions significantly compared to P3. Moreover, user is able to select the transparency level that he/she wants and it is guaranteed that all FMs meet those criteria at any time of optimization. This distinguishes it from those approaches which optimize a transparency index (e.g. [13, 15, 39] and P2).

In order to accomplish that, this publication proposes some novel genetic operators and a novel initialization method which combines the benefits of DT initialization [3] and WM initialization [90]. This initialization method
has several desirable characteristics, such as, it guarantees transparency of fuzzy partitions, it selects the number of input variables and assigns a moderate number of MFs to them.

Results are rigorously compared to several state-of-the-art MGFSs (including the MGFS in P3) and GFSs and it is shown that the accuracy and interpretability of the obtained FMs is always at least at the same level with the comparative methods. Moreover, the proposed MGFS clearly outperforms some of the comparative methods.


This publication modifies the MGFS proposed in P4 so that it can be used in classification problems as well. Results are compared to P1 and it is shown that the same level of accuracy and complexity as in P1 can be obtained with the proposed MGFS. What, however, distinguishes this MGFS from the MGFS in P1 is that it guarantees the transparency of fuzzy partitions whereas the MGFS in P1 does not. Therefore, there is an improvement in interpretability without sacrificing accuracy. This indicates that the MGFS in P5 has good search ability, because it was pointed out in P4 that there is a trade-off between transparency of fuzzy partitions and accuracy. In other words, the MGFS in P5 was expected to obtain worse accuracy than the MGFS in P1 because of this trade-off.

1.3 Contributions of this Thesis as a Whole

The contributions of this thesis as a whole can be summarized in general level as follows. The publications in which these contributions were mainly proposed are indicated in brackets.

1. Development of five MGFSs which can obtain a set of Pareto optimal FMs
presenting trade-offs between accuracy and interpretability. Their application to classification and regression problems. [P1- P5]

2. Development of initialization methods to ease the further optimization by MOEAs. [P3- P5]

3. Development of MFs tuning strategies which take into account transparency of fuzzy partitions. [P2, P4, P5]

4. Industrial application of FM:
   • to develop a bioaerosol detector. [P2]
   • as a reasoning mechanism in a bioaerosol detector. [P2]

1.4 Author’s Contribution

The author of this thesis solely wrote all the publications P1- P5. Moreover, he designed and programmed all the MGFSs and performed all the experiments with them. Hannu Koivisto supervised and proposed corrections to this overview and to publications P1- P4. Jarmo Hytönen gave valuable information about the bioaerosol detector and provided data and two figures of publication P2.

1.5 Organization of this Thesis

In order to give a basic understanding of GFSs and MGFSs, the next Chapter briefly describes FMs followed by EAs and MOEAs which are used to learn the parameters of FMs. The MGFSs in publications of this thesis can be divided into two categories. First category consists of the earlier MGFSs in P1– P3. Each of them has some unique characteristics, however, all of them use 3-parameter MFs tuning with static constraints. That is one of the main differences between them and the MGFSs in P4 and P5, which use 3-parameter MFs tuning with dynamic constraints. Other components of MGFSs in P4 and P5, such as initialization method and chromosome coding, can also be considered more advanced than the corresponding components in MGFSs in P1– P3.
Since the MGFSs in P1–P3 were published first, they are introduced first in Chapter 3 followed by the MGFSs in P4 and P5 in Chapter 4. The obtained results are summarized and discussed in Chapter 5. Finally, conclusions and further research directions are given in Chapter 6.
Chapter 2

Components of Multiobjective Genetic Fuzzy Systems

Genetic fuzzy systems (GFSs) are essentially fuzzy systems optimized by evolutionary algorithms (EAs) [24]. If, instead of EAs, multiobjective evolutionary algorithms (MOEAs) are used to optimize the parameters of fuzzy models (FMs), the approaches are referred to as multiobjective genetic fuzzy systems (MGFSs).

The first GFSs were proposed in 1990s and some of the earlier works are [61, 75, 87, 88]. They were applied to adapt the parameters of FMs in order to improve the accuracy without taking into account the interpretability. Later on the interpretability was also taken into account [1–3, 6, 18, 52, 58, 79, 81, 82], however, in those studies aggregated fitness functions were used. As it has been realized that it is impossible to optimize both objectives simultaneously, the current trend is to use MGFSs instead of GFSs and to obtain a set of non-dominated FMs presenting different trade-offs between accuracy and interpretability [4, 10, 15, 32, 37, 48, 49, 51, 53, 83].

This Chapter presents the main components of GFSs and MGFSs, namely FMs, EAs and MOEAs. After that, the nondominated sorting genetic algorithm II (NSGA-II) [30], the MOEA which is used in this thesis to search for Pareto optimal FMs, is briefly revisited. Finally, the components which affect the size of the search space of MGFSs are discussed and this Chapter is summarized.
2.1 Fuzzy Models

Fuzzy logic has been applied in numerous applications, for example, in time series prediction and modeling nonlinear, uncertain, and complex systems [56]. According to the inventor of fuzzy logic, Lotfi A. Zadeh, fuzzy logic attempts to formalize two human abilities, 1) the ability to reason in the presence of imperfect information and 2) the ability to carry out different mental and physical tasks without any numerical measurements or computations [96]. A key element enabling this is a fuzzy set. In classical boolean logic an object \( x \in U \), where \( U \) is the universe of discourse (UoD), either belongs or does not belong to a set \( B \subseteq U \). However, in fuzzy logic an object \( x \) belongs to a fuzzy set \( B \) to a certain degree between 0 and 1. More formally, a fuzzy set \( B \) is defined as:

\[
B = \{ (x, \mu_B(x)) | x \in U \},
\]

(2.1)

where \( \mu_B(x) \in [0, 1] \) is the membership function (MF) of fuzzy set \( B \). A fuzzy set can have a linguistic value, such as, small, large, young, old etc. For example, if \( U \) is a linguistic variable age (in years), then a person whose age is 55 years may belong to a fuzzy set "Old" with membership degree 0.2 (i.e. \( \mu_{\text{Old}}(55) = 0.2 \)) and to a fuzzy set "Middle aged" with a membership degree 0.8 (i.e. \( \mu_{\text{Middle aged}}(55) = 0.8 \)). That allows smooth transition from one set to another and is useful when dealing with noisy measurements.

Fuzzy sets can be used to form fuzzy "If-then" rules which mimic the way humans reason. Instead of numbers and symbols, humans often operate with concepts of natural language. Fuzzy logic enables computing with words [95], that is, reasoning using fuzzy sets as premises to reach conclusions, which can be either fuzzy or crisp.

A fuzzy inference system (FIS) combines fuzzy set theory, fuzzy rules and fuzzy reasoning. According to [56] a FIS consists of three parts:

1. **Rule base**, which contains the collection of fuzzy rules.
2. **Data base** (or dictionary), which specifies the MFs used in the fuzzy rules.
3. **Reasoning mechanism**, which infers the output based on the rule base, data base and the given input.

FIS accepts fuzzy inputs or crisp inputs, which are commonly fuzzified using fuzzy singletons. The output of FIS is a crisp value when Takagi-Sugeno (TS) type FM [86] is applied and a fuzzy set when Mamdani type FM [66] is used. However, the output of a Mamdani type FM is often defuzzified in order to obtain a crisp output [56].

### 2.1.1 Input and Output Data

This thesis deals with both classification and function estimation problems. Both linear and non-linear problems are considered, however, the emphasis is on non-linear problems. These problems can be either static or dynamic. However, the model structures for presenting the dynamics are limited to finite impulse response (FIR) and to auto regressive external input (ARX) model structures. Other model structures, such as output error (OE) model structure, are not considered in this thesis and an interested reader is referred to [70, 74] for more information about them. If the proposed MGFSs were to be used to identify FMs with OE model structure, at least the fitness evaluation part of MGFSs should be reprogrammed in order to accommodate that. That is an interesting future research topic.

A data set with $D$ data points and $n$ input variables is denoted by $Z = [X \ y]$, where input matrix $X$ and output vector $y$ are given as:

$$
X = \begin{bmatrix}
    x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\
    x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\
    \vdots & \vdots & \ddots & \vdots \\
    x_{D,1} & x_{D,2} & \cdots & x_{D,n}
\end{bmatrix}, \quad
y = \begin{bmatrix}
    y_1 \\
    y_2 \\
    \vdots \\
    y_D
\end{bmatrix}.
$$

(2.2)
The purpose is to infer based on data point $x_k = \left[ x_{k,1} \ x_{k,2} \ldots \ x_{k,n} \right]^T$ the corresponding output $y_k$, where $k = 1, \ldots, D$. Each $x_{k,j}$, $j = 1 \ldots, n$ is a numerical value either continuous or discrete. Each $y_k$ is an integer presenting the class value when classification problems are considered. The class values can be either ordinal or nominal. When regression problems are considered, $y_k$ is a real number.

Publications P1, P2, and P5 deal with classification problems and use Winner-Takes-All as a reasoning mechanism. Publications P3 and P4 deal with regression problems and apply Mamdani FM because its consequent is a linguistic value and therefore interpretability of the rules is at a good level.

### 2.1.2 Gaussian Bell Membership Functions

There are several MFs in the literature and commonly triangular, trapezoidal, Gaussian or Gaussian Bell (gbell) MFs are used. Throughout this thesis gbell input MFs are used:

$$
\mu(x; a, b, c) = \frac{1}{1 + \left| \frac{x - c}{a} \right|^2},
$$  

(2.3)

where $x \in U$ is a data point, and $a$, $b$, and $c$ are the parameters of a gbell MF. Parameter $a$ defines the width of an MF. By increasing its value, the MF becomes wider and by decreasing its value, it becomes more narrow. Parameter $b$ defines the fuzziness of an MF. If it is set to a high value, an MF resembles classical crisp set. On the other hand, if it is set to low value an MF becomes more fuzzy and covers large areas of UoD. Finally, parameter $c$ defines the center of an MF.
2.1.3 Fuzzy Rules

Fuzzy rules are expressed in the following form:

\[ R_i : \text{If } x_1 \text{ is } B_{i,1} \ldots \text{ and } x_n \text{ is } B_{i,n} \text{ then } C_i, i = 1, \ldots, R, \quad (2.4) \]

where \( R \) is the number of rules, \( B_{i,j} \), \( j = 1, \ldots, n \) is an input fuzzy set and \( C_i \) is the rule consequent. For classification problems, \( C_i \in \{1, \ldots, n_{\text{Classes}}\} \), where \( n_{\text{Classes}} \) is the number of different classes in the data set. If regression problems are considered, \( C_i \) is an output fuzzy set. Regardless of the problem type, the rule fulfillment degree is computed for each data point \( x_k \) using product t-norm:

\[ \beta_i(x_k) = \prod_{j=1}^{n} \mu_{B_{i,j}}(x_{k,j}). \quad (2.5) \]

There are other types of fuzzy rules and t-norms which can be applied to reasoning [23] and the properties of fuzzy classifiers are discussed in detail in [64]. Outcome \( \hat{y}_k \) of an FM depends on the problem type. For classification problems, the rule with the highest rule fulfillment degree is declared as the winner rule (i.e. Winner-Takes-All strategy) and \( \hat{y}_k \) is the rule consequent associated to that rule [3]:

\[ \hat{y}_k = C_{i^*}, \text{ where } i^* = \arg \max_{1 \leq i \leq R} \beta_i(x_k). \quad (2.6) \]

If the problem type is regression, the output needs to defuzzified. Some of the defuzzification methods, such as the conventional centroid of gravity defuzzification method [56], are computationally demanding. Since MOEAs are applied to optimize FMs, defuzzification is performed whenever fitness is evaluated. Thus, it is beneficial to use a computationally less demanding defuzzification method. Therefore, \( \hat{y}_k \) is computed as approximation of centroid of gravity method [19,97],
which reduces the computationally costs:

\[
\hat{y}_k = \frac{\sum_{i=1}^{R} \beta_i(x_k) \tilde{C}_i}{\sum_{i=1}^{R} \beta_i(x_k)},
\]

(2.7)

where \( \tilde{C}_i \) is the center value of an output fuzzy set \( C_i \). Since \( \tilde{C} \) is the only output MF parameter affecting the outcome, application of singleton output MFs is a natural choice:

\[
\mu(x, \tilde{C}) = \begin{cases} 
1, & \text{if } x = \tilde{C} \\
0, & \text{if } x \neq \tilde{C} 
\end{cases}
\]

(2.8)

### 2.2 Evolutionary Algorithms

EAs belong to the field of evolutionary computation and comprise of different population-based stochastic (non-deterministic) algorithms, such as, genetic algorithms (GAs), evolution strategies (ES), evolution programming (EP) and genetic programming (GP). All of them can also be referred as "Generate-and-Test" algorithms, because they generate offspring in a population using genetic operators mutation and/or crossover. Then, commonly the population for the next generation is selected among offspring and their parents based on each individuals’ fitness (i.e. their fitness is tested). [94] There are differences among GA, ES, EP, and GP, mainly in chromosome presentation and in genetic operators. For example, a traditional GA presents a chromosome using bit strings, whereas EP presents a chromosome using real-coded vectors [67, 94]. Moreover, a GA uses both mutation and crossover, whereas an EP uses only mutation. However, since they share more common factors than they have differences, a more general name, EA, is often used [94]. This name is also used in this thesis.

EAs perform exploration and exploitation by the genetic operators, mainly mutation and/or crossover. Exploration of the search space adds randomness to the
search and makes it possible for EAs to escape from a local minimum. The current solutions are exploited in order to find better solutions characterized by good fitness values. Computation of derivatives is not required but the search is guided by a selection mechanism, which favors solutions with good fitness values. Moreover, elitism, that is retaining the best solution found so far, is applied in most of the recent EAs. Because of these characteristics, EAs can be called guided random search algorithms. The pseudo code of an EA can be given as:

1. Initialize the population.
2. Evaluate fitness of each individual of the initial population.
3. Repeat the following for $G_{\text{tot}}$ iterations or until another stopping criteria is met:
   (a) Select individuals for reproduction.
   (b) Perform the genetic operators.
   (c) Evaluate fitness of each individual of the offspring population.

So EAs maintain a population of candidate solutions, which are often called as individuals, chromosomes or genotypes [67]. When EAs are used to optimize the parameters of FMs, a chromosome is an FM when a Pittsburgh style [84] EA is considered and it can be a single rule when Michigan style [43] EA is considered. Each genotype has a specified meaning – phenotype. For example, if a fuzzy rule selection problem with 5 candidate rules is to be solved using an EA, a genotype ‘1 0 1 0 0’ can mean that two fuzzy rules (the first and the third) are selected. This is because the values of the first and the third genes are ‘1’ [54]. The mapping from genotype to phenotype is always defined a priori by user. In the fuzzy rule selection problem, the candidate rules also need to be stored for example in the memory and sometimes they are also optimized during an EA run.

Population is usually initialized randomly or using some heuristic algorithm. Random initialization usually means that the user specifies which variables are to be optimized and the range they can vary in. Then, the values of the variables are assigned to random values between their constraints. Heuristic algorithm, on the other hand, uses some heuristics to initialize the values of variables. The purpose
is to obtain a better starting point for further optimization than with random initialization. It is also possible and often preferred that some heuristics are used to reduce the number of variables to be optimized [3]. In this thesis decision tree (DT) based heuristic algorithms are used to initialize the population mainly because they ease the further optimization due to reduction in the search space.

The size of the search space is mainly defined by the number of variables to be optimized and by their coding scheme. Sometimes the problem itself determines how many variables are to be optimized. This is usually the case when a mathematical function with certain amount of variables is to be optimized. In other cases, such as, in the above fuzzy rule selection problem, the number of variables to be optimized can be controlled by the user who can, for example, determine the maximum number of candidate rules.

The variables can be coded in several ways and not every variable needs to coded the same way. The applied coding scheme affects the design of constraint handling, fitness evaluation and the genetic operators. Binary coding of the variables is the most conventional way and well suited for certain problems. However, in some problems other coding schemes, such as, integer or real coding are more appropriate [67].

The fitness function defines what kind of solutions are regarded as fit. The characteristics which are either directly or indirectly connected to the fitness function will be subject to optimization. Naturally, the characteristics which are not connected to the fitness function in any way, will not be taken into account. Selection mechanism then selects based on the fitness values the solutions which are to be preserved and which are to be deleted. Various selection mechanisms exist and some well known mechanisms are roulette wheel selection, ranking selection and tournament selection [67].

Genetic operators, mainly crossover and mutation, are used to alter the genes in order to find fitter solutions to the problem. Usually mutation is more or less a small random change in one or more of the variables of one solution. Crossover, on the other hand, involves two solutions often called parents. They produce one or more solutions called offspring, which inherit the genetic material from the parents. When dealing with problems with large search spaces, appropriate genetic operators become very important for the convergence of EAs.
Finally, standard EAs start with an initial population of chromosomes (FM in this thesis) and end up with the same kind of chromosomes, which in most cases are fitter than the ones in the initial population. For instance, when standard GFSs are considered, FMs do not evolve to any other model type no matter how many generations they are run. Instead, their parameters are adjusted by genetic operators leading to fitter FMs. Evolution from FM to another model type is technically possible but requires that GFS is provided with necessary information about the model type(s) FM can evolve to. Thus, many parts of the standard GFS, such as chromosome coding, genotype to phenotype mapping, fitness evaluation, and genetic operators, should be reprogrammed to accommodate that. To the best knowledge of the author of this thesis, there exists no such GFS in the literature.

2.3 Multiobjective Evolutionary Algorithms

MOEAs resemble EAs, however, the major distinction is that MOEAs optimize simultaneously several objectives, whereas EAs optimize only one objective. Therefore, the fitness function in EAs is a scalar and in MOEAs a vector of length \( h \), where \( h \) is the number of objectives. Due to that, the selection mechanisms are also different. When the fitness function is a scalar, the best solution is simply the one having the smallest fitness value if the problem is of minimization type. Likewise, if the problem is of maximization type, then the solution with largest fitness value is the best solution. Naturally, a maximization problem can be solved as a minimization problem by multiplying the fitness value by -1.

When two or more objectives are to be optimized simultaneously, the problem is referred as multi-objective problem (MOP). Let \( f_i, i = 1, \ldots, h \) be the \( h \) objectives to be optimized and let \( s \) be a solution and \( S \) the feasible region in the search space. Then, MOP can be formulated as a minimization problem as follows [21], [51]:

\[
\text{Minimize } f_1(s), f_2(s), \ldots, f_h(s) \text{ subject to } s \in S. \quad (2.9)
\]

Usually it is not possible to obtain a solution which simultaneously minimizes all \( h \) objectives. However, it is possible to find a set of non-dominated (Pareto
optimal) solutions. A solution \( s_1 \in S \) is Pareto optimal, if there is no solution \( s_2 \in S \), which fulfills the following conditions:

\[
\forall i, f_i(s_2) \leq f_i(s_1) \quad \text{and} \quad \exists j, f_j(s_2) < f_j(s_1).
\]  

(2.10)

If \( s_2 \) meets the conditions in (2.10), it dominates \( s_1 \). A conventional approach for solving a MOP is to aggregate the objectives into one objective using some weights \( w_1, w_2, \ldots, w_h \):

\[
\text{Minimize } w_1 f_1(s) + w_2 f_2(s) + \ldots + w_h f_h(s) \text{ subject to } s \in S.
\]  

(2.11)

By doing so, a MOP is transformed into a single-objective problem, which can be solved using EAs or some other single-objective optimization technique. As a result a single solution is found. It is, however, not known whether a more suitable solution had been found if different weights would be used. Therefore, several runs with different weights are usually required. Moreover, the weights may not be the same for different problems. Also, if an EA is run several times with different weights, it is not guaranteed that with every run a new solution is found [30]. Furthermore, EAs cannot obtain those solutions which are situated in the concave region of the Pareto front [60]. Finally, it has been reported that MOEAs can often find better trade-off FMs than EAs [37].

When MOEAs are used, the user is not required to set any weights and with a single run multiple Pareto optimal solutions are found. These non-dominated solutions form so called Pareto optimal set, which is often plotted under the objective functions in order to select a suitable solution for a problem or to compare the quality of solutions obtained by different MOEAs. These figures are called Pareto frontiers or Pareto fronts and an example is illustrated in Fig. 2.1. In this Figure all solutions are FMs presenting different trade-offs between accuracy and interpretability. Selecting an FM depends on the problem and the user preferences. The advantage of using MOEAs is obvious. Instead of specifying the weights beforehand and running the EA multiple times, the user can now select a solution after he/she has seen the available choices. For example, it might be reasonable to choose a solution which has about 40 rule conditions instead of a solution with
Figure 2.1: An example Pareto front of a minimization type problem. The purpose is to minimize the number of rule conditions and training MSE. The obtained FMs present trade-offs between these objectives and the user selects one or more of them according to his/her preferences. The data set is Abalone and will be introduced in Chapter 5.

Due to these reasons, MOEAs have become popular in solving various problems [51, 63, 98]. Before the first implementations of MOEAs in mid-1980s, MOPs have been solved in the field of operations research since late 1950s using different mathematical programming methods [21]. Some of these methods, however, are suitable only for differentiable problems [68], which is a severe limitation with regards to solving many real-world problems. Furthermore, some of these methods work only for problems which do not have concave or disconnected Pareto fronts [21]. Finally, multiple runs from different starting points are usually

more than 70 rule conditions, because the difference in mean squared error divided by two (MSE) between these solutions is very small. So, improvement in interpretability due to smaller rule base causes only a small deterioration in accuracy, which may be acceptable. On the other hand, when number of rule conditions is reduced from 28 to 26, the small interpretability improvement causes a large increment in MSE, which may not be worthwhile.
required in order to obtain multiple Pareto optimal solutions [21, 45, 68]. Since MOEAs more or less overcome these problems they are used in this thesis. There exist several good MOEA implementations, such as, the strength Pareto evolutionary algorithm 2 (SPEA2) [98], the Pareto archived evolution strategy (PAES) [65] and the the nondominated sorting genetic algorithm II (NSGA-II) [30].

This thesis applies NSGA-II, which was simply selected because it led to good results in the preliminary tests. It is, of course, possible that by applying another MOEA the results could be improved. Studying the performance of MGFSs using different MOEAs when applied to optimize the parameters of FMs but good results are reported for NSGA-II [16, 51], PAES [7, 19], and SPEA2 [37, 39]. Hence, it appears to be more important to design, for example, appropriate genetic operators, coding scheme, MFs tuning strategy, and initialization method than to choose a particular MOEA. The next Section briefly outlines the NSGA-II.

### 2.4 NSGA-II: The Nondominated Sorting Genetic Algorithm II

NSGA-II was developed by Deb et al. [30]. It has become a very popular MOEA because it overcomes the three limitations of NSGA [85], namely lack of elitism, high computational demand, and the requirement to set a sharing parameter in order to maintain diversity of population. The computational complexities of NSGA-II and NSGA are respectively $O(hN_{\text{pop}}^2)$ and $O(hN_{\text{pop}}^3)$, where $h$ is the number of objectives and $N_{\text{pop}}$ is the population size. NSGA-II utilizes elitism mechanism and diversity of population is maintained by parameterless crowding distance selection. Its source code in C-language is freely available from: [http://www.iitk.ac.in/kangal/codes.shtml](http://www.iitk.ac.in/kangal/codes.shtml).

NSGA-II starts with an initial population of size $N_{\text{pop}}$. The chromosomes are sorted using the fast nondominated sorting algorithm. Based on this, a rank for each chromosome is assigned. The chromosomes with rank 1 are not dominated by any chromosome. Chromosomes with rank 2 are dominated by the chromo-
Components of Multiobjective Genetic Fuzzy Systems

Components of Multiobjective Genetic Fuzzy Systems

somes with rank 1 and chromosomes with rank 3 are dominated by the chromo-
somes with ranks 1 and 2. So, the chromosomes with lower rank dominate the
chromosomes with higher rank. From the current population of size \( N_{\text{pop}} \), the off-
spring population of size \( N_{\text{pop}} \) is created by binary tournament selection, crossover
and mutation. The binary tournament selection is based on rank. However, the
chromosomes presenting the same rank are selected based on crowding distance,
such that, the chromosomes which are located at the less crowded part of the ob-
jective space are preferred to chromosomes which are located at the more crowded
part of the objective space. This promotes diversity among the chromosomes and
makes it possible to obtain widely distributed set of Pareto optimal chromosomes.

NSGA-II preserves the best chromosomes (i.e. elitism) by following selection
mechanism. First, the current and the offspring populations are combined together
as one population of size \( 2N_{\text{pop}} \). Then, this population is sorted using the fast
nondominated sorting algorithm. The chromosomes with rank 1 are first included
into the new population of size \( N_{\text{pop}} \). Then, if there is still space, the chromosomes
with ranks 2, 3 and so on are included. This is continued until chromosomes with
some rank do not totally fit into the new population. To fill up the new population,
some of these chromosomes are selected based on their crowding distance. [30,51]

Constraints are handled, such that, when two feasible chromosomes are compared,
the one which is superior based on rank (or based on crowding distance if their
rank is the same) is selected. If one chromosome is feasible and the other infea-
sible, the feasible chromosome is selected. If both chromosomes are infeasible,
the one with smaller constraint violation is selected [30]. One may notice that this
constraint handling method requires no parameters to be set by user.

2.5 Multiobjective Genetic Fuzzy Systems

The procedure of data-driven identification of FMs by MGFSs is shown in Fig.
2.2. The main steps are initialization and further optimization by MOEA. When-
ever a new MGFS is proposed in the literature, usually the purpose is to improve
the initialization and/or the further optimization parts. As an initialization algo-

rithm, any algorithm which identifies FMs based on data in a relatively short time
can be used. However, a good initialization algorithm creates the population, such
that, the further optimization by MOEA is made easier. This can mean, for example, that irrelevant input variables are removed and the remaining input variables are partitioned with reasonable number of fuzzy sets. On the other hand, the performance of further optimization by MOEA depends on various factors, such as, chromosome coding, genetic operators, and the MOEA that is used.

In the following, some common initialization methods are presented. Then Section 2.5.2 explains how the number of parameters to be optimized depends on which of the FMs components are subject to optimization. Finally, Section 2.5.3 discusses the coding of FMs and shows that this aspect has an effect on the size of the search space as well.

2.5.1 Some Common Initialization Methods

Each MGFS needs to be initialized. That can be done manually, randomly or using a heuristic algorithm. Practically none of the current approaches use manual initialization because it is a tedious task to set all parameter values manually. In this Section the benefits and drawbacks of random initialization, clustering, DT initialization [3], and Wang-Mendel (WM) algorithm [90] are discussed and their characteristics are summarized in Table 2.1. These are some of the commonly used initialization methods and they are discussed here in rather general level. However, the author is aware that, for example, random initialization and clustering can be implemented in many different ways.

Figure 2.2: Data-driven identification of FMs by MGFSs.
Table 2.1: Comparison of initialization methods. \( R \) and \( R_{\text{cond}} \) respectively stand for the number of rules and the total number of rule conditions, whereas \( n \) is the number of input variables and \( M_{\text{max}} \) the maximum number of input MFs in each input variable. Finally, \( N_{\text{clusters}} \) is the number of clusters and MFs equals to the total number of input MFs.

<table>
<thead>
<tr>
<th>Method</th>
<th>( R )</th>
<th>( R_{\text{cond}} )</th>
<th>MFs</th>
<th>Input variable</th>
<th>Transparency of selection</th>
<th>Transparency of fuzzy partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>Varies</td>
<td>Varies</td>
<td>( M_{\text{max}} \times n )</td>
<td>No</td>
<td>Usually transparent</td>
<td></td>
</tr>
<tr>
<td>Clustering</td>
<td>( N_{\text{clusters}} ) ( \times R ) ( \times n )</td>
<td>( R \times n )</td>
<td>( R \times n )</td>
<td>No</td>
<td>Varies</td>
<td></td>
</tr>
<tr>
<td>DT</td>
<td>Varies</td>
<td>Varies</td>
<td>Varies</td>
<td>Yes</td>
<td>Varies</td>
<td></td>
</tr>
<tr>
<td>WM</td>
<td>Varies</td>
<td>( R \times n )</td>
<td>( M_{\text{max}} \times n )</td>
<td>No</td>
<td>Usually transparent</td>
<td></td>
</tr>
</tbody>
</table>

Random initialization is commonly used to randomly set the parameter values between the parameter’s lower and upper limits. This is a simple and fast way to initialize FMs and it can be easily designed, such that, transparency of fuzzy partitions is guaranteed. For example, each input variable can be partitioned with \( M_{\text{max}} \) equally distributed uniformly shaped MFs and only the rules are randomly created [19]. The maximum number of input MFs per input variable, \( M_{\text{max}} \), is an integer and usually between 3 and 9. Of course, not every input variable need to have the same amount of MFs but that is usually the case unless some a priori knowledge is available. This method has some drawbacks: 1) input variables are not selected and 2) starting point may vary greatly due to randomness (i.e. lack of robustness). Finally, 3) the number of MFs for each input variable and often also their parameters need to be provided by user.

Clustering is another way to initialize the population. It can be used to create rules and partition the variables [1]. Its limitations are: 1) the number of clusters, \( N_{\text{clusters}} \), needs to determined by user and that number equals to the number of rules, \( R \), and the number of MFs in each input variable of initial FM. There, however, exists various methods, such as Xie and Beni’s index [93], for determin-
Chapter 2

ing the number of clusters automatically. Other drawbacks are: 2) input variables are not selected and 3) for each rule, the number of rule conditions, $R_{\text{cond}}$, equals to the number of input variables $n$. Finally, 4) fuzzy partitions are not always transparent, but may contain highly overlapping fuzzy sets.

DT initialization creates a DT and converts it to an FM. Its benefits are that it selects the input variables and partitions the input variables with flexible amount of FMs (i.e. not every input variable needs to be partitioned with the same amount of MFs). Also, for some data sets it creates a compact set of rules which do not need to have the same number of rule conditions in each rule. Its limitations are: 1) it creates too many rules for data sets with high noise level and 2) the fuzzy partitions are not always transparent, but may contain highly overlapping MFs and/or the number of MFs for some input variables can be high.

Because WM algorithm creates rules for fuzzy partitions provided by user, the fuzzy partitions are usually transparent and the number of MFs is moderate (usually between 3 and 9). Its limitations are: 1) it may lead to a large number of rules and rule conditions for high-dimensional problems and/or problems with many data points. That is because the rules are created for each data point by finding for each input variable the MF which matches the best with the data point. Therefore, each data point can potentially create a rule. Nonetheless, duplicate rules are not allowed, which reduces the number of rules. 2) Input variables are not selected and each rule has as many rule conditions as there are input variables. 3) User needs to provide the number of MFs and their parameters for each variable.

It can be concluded from this discussion that none of these initialization methods has all of the following desired characteristics: robustness, ability to create a moderate number of rules and rule conditions, ability to select the input variables, and ability to create transparent fuzzy partitions with moderate number of MFs. In this thesis, DT initialization is used in publications P1–P3 (Chapter 3) mainly because of its input variable selection capability. On the other hand, in P4 and P5 (Chapter 4) the benefits of DT and WM methods are combined. This combination has several desirable properties: fuzzy partitions are transparent and consist of moderate number of MFs, input variables are selected, and the number of rules and rule conditions is usually moderate. Its robustness for data sets with high level of noise, is probably its weakest link. For those data sets, it tends to create rather large amount of rule conditions.
2.5.2 The Number of Parameters to be Optimized

MGFSs are very flexible and the genetic operators can be designed, such that, any part of FMs, including the fuzzy inference [27] [9] and defuzzification method [25], can be optimized by them. Moreover, input variable selection, rule selection, fuzzy partition and rule generation can be performed by them [45]. In fact, flexibility is one of the main advantages of MGFSs compared to neuro-fuzzy systems, such as, adaptive-network-based fuzzy inference system (ANFIS) [55].

The performance of MGFSs is highly affected by which parts of FMs are learnt. This, on the other hand, partially determines the size of the search space. In this thesis rule base and data base are learnt by MOEA and this Section discusses the factors which affect the search space in this case. Furthermore, it explains how these factors are handled in this thesis and in the literature.

Suppose the purpose is to learn the rule base and data base for a problem with \( n \) input variables and with one output variable. Let us assume that the number of fuzzy rules is at most \( R_{\text{max}} \) and the maximum number of MFs in each variable \( j \) is at most \( M_{\text{max}} \). If for each rule \( n \) antecedent conditions and one consequent condition are to be learnt, the whole rule base may be described with \((n + 1)R_{\text{max}}\) parameters. If each MF is described with \( \rho \) parameters, then \((n + 1)\rho M_{\text{max}}\) parameters are needed to define the data base when regression problems are considered. For classification problems there are no MFs assigned to the output so the data base is presented with \( n\rho M_{\text{max}} \) parameters. Thus, the number of parameters required to define knowledge base for regression and classification problems is given as:

\[
\theta_{KB} = \begin{cases} 
(n + 1)(R_{\text{max}} + \rho M_{\text{max}}) & \text{for regression problems} \\
(n R_{\text{max}} + \rho M_{\text{max}}) + R_{\text{max}} & \text{for classification problems}
\end{cases}
\]

One may notice that even if \( n = 9, R_{\text{max}} = 10, \rho = 3, M_{\text{max}} = 5 \), the number of parameters to be optimized is already \((9 + 1)(10 + 15) = 250\) for regression problems and \(9 \times (10 + 15) + 10 = 235\) for classification problems. Those are rather typical values and it is common for \( n \) to be much larger. If, however, only the rule
base is optimized, the number of parameters to be optimized is reduced to 100 for both problem types. On the other hand, if only the data base is optimized, the number of parameters to be optimized is 150 for regression problems and 135 for classification problems. That is one of the main motivations why in some studies only rule base or data base is optimized. However, simultaneous optimization of both rule base and data base enables finding more suitable FMs. One must, therefore, try to minimize $n$, $R_{\text{max}}$, $\rho$, and $M_{\text{max}}$ in order to reduce the search space. In the following, some of the common attempts are briefly presented.

The attempt to reduce $n$ seems promising as it directly affects the number of parameters required to define both rule and data base. Surprisingly, most of the current MGFSs pay no attention to minimize $n$ by removing irrelevant input variables in the initialization phase. In all publications of this thesis, however, DT based initialization method is applied in an attempt to remove irrelevant input variables.

It is common to restrict $R_{\text{max}}$ to an integer which is usually not more than 100 [7, 11, 19, 51]. The common values for $R_{\text{max}}$ are 30 and 40. Restricting $R_{\text{max}}$ is reasonable because very large rule bases would be difficult to interpret in any case. Another way is to use WM algorithm to initialize the population and use the number of rules it created as $R_{\text{max}}$ [4, 37]. That, however, leads to large number of rules for some data sets. In P1, P2 and P3, $R_{\text{max}}$ was the same as the number of rules in the initial FM created by DT initialization. In some cases, however, that number can be rather large and therefore in P4 and P5 $R_{\text{max}}$ is at most 30.

The value of $\rho$ depends on the type of MFs that are used and on the tuning strategy. If MFs are not tuned at all, $\rho$ is naturally zero like in [51]. If MFs are tuned and triangular or gaussian bell (gbell) MFs are used, then $\rho = 3$. On the other hand, if gaussian MFs are used, $\rho = 2$. In order to minimize $\rho$, linguistic 2-tuple representation scheme has been proposed in [42]. That scheme reduces $\rho$ to one, since it alters only the lateral displacements of MFs. However, the shape of the MFs is restricted, which means that the obtained FMs are usually not as accurate as when more relaxed MFs tuning scheme is used. Linguistic 3-tuple representation [4], on the other hand, alters also the width of the MFs and therefore $\rho = 2$. Thus, it can lead to better accuracy than 2-tuple representation. However, the FMs obtained by 3-tuple MFs tuning are usually considered less interpretable due to deterioration in the transparency of fuzzy partitions [4]. In this thesis 3-
parameter MFs tuning is applied so \( \rho = 3 \). That was selected in order to obtain accurate FM s. Transparency of fuzzy partitions is ensured in P2 by minimizing a transparency index and in P4 and P5 by dynamic constraints. In P1 and P3 transparency of fuzzy partitions is not guaranteed.

In [51] MFs were taken from four different partitions with 2, 3, 4, and 5 MFs, so the maximum number of MFs, \( M_{\text{max}} \), in a global fuzzy partition can be at most 14 (i.e. 2 + 3 + 4 + 5). In other works [4, 7, 17, 37], the value of \( M_{\text{max}} \) is usually set to 3, 5, 7, or 9 by user a priori. For higher dimensional problems, \( M_{\text{max}} \) is usually set to 3 or 5 [7, 17]. It is also possible that \( M_{\text{max}} \) for input variables is lower than \( M_{\text{max}} \) for the output variable [17]. It should be also noticed that equal number of MFs per variable usually does not lead to very good accuracy. Moreover, different learning methods may obtain the best results with different granularities and for different problems different granularities are usually required [26].

Therefore, the granularity of each variable should not be fixed but learnt for each problem. That is done in each publication of this thesis, such that, only the maximum number of MFs (\( M_{\text{max}_j} \)) per variable is constrained but FM s can have less than \( M_{\text{max}_j} \) MFs in input and output variables. In P1, P2, and P3, \( M_{\text{max}_j} \) for each variable is determined by the number of MFs in initial FM (\( M_{\text{DT}_j} \)). On the other hand, in P4 and P5, it is determined as a minimum of a user specified value \( M_{\text{in}} \) and \( M_{\text{DT}_j} \). Usually \( M_{\text{in}} \) is between 3 and 9. Notice that in all publications of this thesis \( M_{\text{max}_j} \) does not need to be the same for all \( j \).

Table 2.2 summarizes the typical values of \( R_{\text{max}} \), the number of input variables, \( \rho \), and \( M_{\text{max}_j} \) in the literature [4, 7, 11, 37, 51] and in this thesis. The approaches included here are some of the recent MGFSs for learning linguistic FM s. They perform MFs tuning and/or rule learning (or rule selection).

### 2.5.3 Coding of Fuzzy Models

The FM’s components, such as rules and parameters of MFs, need to be presented such that MOEA can be used. This Chapter discusses what kind of coding is usually used for different FM’s components. Moreover, an example of coding an FM in case when rule learning and MFs tuning is performed is shown.
Table 2.2: Common parameter values in some MGFSs in the literature [4, 7, 11, 37, 51] and in the publications of this thesis. $M_{\text{max}_j}$ stands for the maximum number of input MFs in input variable $j$, where $j = 1, \ldots, n$.

<table>
<thead>
<tr>
<th>Reference</th>
<th>$R_{\text{max}}$</th>
<th>Input variables</th>
<th>$\rho$</th>
<th>$M_{\text{max}_j}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commonly in literature</td>
<td>[20, 100] or $n$</td>
<td>{0, 1, 2, 3}</td>
<td>$M_{\text{max}}$ for all $j$, usually [3, 14]</td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>Determined by DT</td>
<td>$\leq n$, 3</td>
<td>$M_{\text{DT}_j}$</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td></td>
<td>30</td>
<td>$\leq n$, 3</td>
<td>$\min(M_{\text{DT}<em>j}, M</em>{\text{in}})$</td>
</tr>
<tr>
<td>P5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Like mentioned, binary-coding is commonly used to indicate whether a particular rule is included or excluded in a rule selection problem. On the other hand, when rule learning is performed, integer-coding is often used. Consider the following regression problem with two rules, two input variables ($x_1$ and $x_2$) and with one output variable:

Rule 1: If $x_1$ is small and $x_2$ is large, then output is small

Rule 2: If $x_1$ is large and $x_2$ is small, then output is large

Let us denote linguistic values small and large (see Fig. 2.3) by integers 1 and 2, respectively. Moreover, denote "don’t care" condition (i.e. an input variable is ignored in the rule) by 0. Also, assume that $M_{\text{max}_1} = M_{\text{max}_2} = 2$, which means
that there can be at most two MFs assigned to both input variables. Then, the antecedents and consequents can respectively be represented for rule learning as:

\[
A = \begin{bmatrix}
1 & 2 \\
2 & 1
\end{bmatrix}
\]

and

\[
s = \begin{bmatrix}
1 \\
2
\end{bmatrix},
\]

where the first and second row correspond the rule conditions of the first and second rule, respectively. In this case, binary representation could also be used. Two bits would be enough to present 0 (i.e. 00), 1 (i.e. 01), and 2 (i.e. 10). However, the binary number 11 corresponds to 3 but since \( M_{\text{max}_1} = M_{\text{max}_2} = 2 \), this would be an infeasible solution. That should be prevented by genetic operators or by repair operators. It is, however, more practical to use integer-coding.

Real-coding (i.e. rule conditions are presented with floating point numbers) could also be used. It, however, increases the search space dramatically. When integer-coding is used, there are only three possible values for each rule condition (i.e. 0,
However, when real-coding is used, there are many\(^1\) possible values for each rule condition. Also, if genetic operators cause a rule condition to be modified, say from 1 to 1.43, it would be usually rounded to the nearest integer which is 1. So, as a result of applying genetic operators, the same solution as before is obtained which is a waste of computational resources. The input and output MF parameters are usually presented as real-coded matrices \(P\) and \(O\), respectively:

\[
P = \begin{bmatrix}
0.20 & 2.50 & 0.00 \\
0.80 & 9.00 & 1.00 \\
0.35 & 2.00 & 0.00 \\
0.60 & 7.00 & 1.00 \\
\end{bmatrix}
\quad \text{and} \quad
O = \begin{bmatrix}
0.30 & 5.00 & 0.00 \\
0.70 & 5.00 & 1.00 \\
\end{bmatrix}
\]

In these matrices, each row contains the parameters \(a\), \(b\), and \(c\) of a gbell MF, respectively. The first two rows of \(P\) stand for the MFs of \(x_1\) and the last two rows for the MFs of \(x_2\). If singleton output MFs are used, the output MF parameters can be presented with real-coded vector \(o = [0.00, 1.00]^T\) instead of \(O\).

### 2.6 Summary

This Chapter presented the main components of GFSs and MGFSs, namely FMs, EAs, and MOEAs. It also presented some common initialization methods. Furthermore, it was shown that various possibilities of optimizing the parameters of FMs exist and that by selecting different parts of FMs as parameters to be optimized, the size of the search space is highly affected. Moreover, it was pointed out that the search space is also affected by the coding of the parameters.

The following two Chapters will present the details of the five MGFSs proposed in this thesis. The next Chapter presents the MGFSs which were proposed earlier.

---

\(^1\)The amount depends on how many bits are used for representing a floating point number.
in P1–P3. They are characterized by the usage of 3-parameter MFs tuning with static constraints, whereas the later MGFSs in P4 and P5 use 3-parameter MFs tuning with dynamic constraints. They are presented in Chapter 4.
Chapter 3

Proposed Multiobjective Genetic Fuzzy Systems with Static Constraints (P1–P3)

This thesis proposes five multiobjective genetic fuzzy systems (MGFSs), which implement the initialization and further optimization parts differently. This chapter discusses the implementation of the first MGFS proposals in P1–P3. They are characterized by 3-parameter membership functions (MF) tuning with static constraints and the usage of the genetic operators of nondominated sorting genetic algorithm II (NSGA-II) without any modifications. Another common factor is that they use real-coded variables to present the rule base. Finally, their initialization methods resemble each other much more than they resemble the initialization methods of the second MGFS proposals in P4 and P5 (introduced in Chapter 4).

3.1 Initialization Methods

The initialization methods are based on decision tree (DT) initialization proposed in [3]. The main idea is to create a DT using C4.5 [78] algorithm and to transform it into a fuzzy model (FM) according to [3]. In that paper, the initial FMs were further optimized by evolutionary algorithm (EA) in order to improve the accuracy
and interpretability. The conflicting objectives were aggregated using weights leading to a single objective to be optimized. Thus, a set of Pareto optimal FMs was not obtained and the user needed to set weights. Moreover, the initialization method in that paper was applicable only to classification problems. In this thesis, DT initialization [3] is used together with multiobjective evolutionary algorithm (MOEA), so a set of Pareto optimal FMs is obtained by a single run and setting of weights is not required. P1 is probably the first publication in the literature, which uses this initialization method with MOEA. Moreover, this initialization method is modified in P3 to be applicable for regression problems as well.

The following two Sections summarize the application of this initialization method for classification and regression problems, respectively. In these Sections, only the creation of the first FM of the initial population is discussed. The rest of the initial FMs are created by randomly modifying the parameters of the first FM, such that, the parameters of initial population are widely spread. The details of this operation can be found in P1.

### 3.1.1 Initialization Method for Classification Problems

In P1, the DT initialization was used almost as it is. The only difference was the application of gbell MFs instead of trapezoidal FMs. That reduced the number of parameters per MF from four to three, which reduced the search space. This initialization method does not guarantee transparency of fuzzy partitions and therefore the subsequent publications P2 and P3 applied merging of fuzzy sets and removal of inconsistent rules and rule conditions as an additional step. That improved the obtained FMs in terms of compactness and transparency of fuzzy partitions. This improvement, however, usually deteriorated the accuracy of initial models significantly. Moreover, transparency of fuzzy partitions, although improved, usually still required further improvement, which was achieved in P2 by optimizing a transparency index during MOEA optimization.
Proposed MGFSs with Static Constraints (P1–P3) 39

Figure 3.1: The initialization method for regression problems. For classification problems, step 1 is skipped.

3.1.2 Initialization Method for Regression Problems

Usually when regression problem are considered, the initial population is created either randomly or by WM algorithm [90]. Both of these methods have their drawbacks, which were discussed in Section 2.5.1. Encouraged by the performance of C4.5 initialization method in classification problems, it was interesting to study whether it would lead to good performance in regression problems as well. However, C4.5 is applicable only to classification problems. Therefore, in P3 the output data were discretized before applying C4.5 algorithm in order to present the continuous values as classes (see Fig. 3.1). That was done by dividing the output to $M_{\text{out}}$ crisp regions, where $M_{\text{out}}$ is a user given positive integer (usually between 3 and 9). Each output value belongs to one of these $M_{\text{out}}$ regions and it is replaced with a corresponding class label, representing these regions. After that, a DT can be constructed and transformed into an FM. However, because regression problems are considered, the output of FM is not a class label but an MF. The output MF center $\tilde{C}_i$ for rule $i = 1, \ldots, R$ is denoted as:

$$\tilde{C}_i = c_{s_i},$$  \hspace{1cm} (3.1)
where $s_i \in \{1, \ldots, M_{\text{out}}\}$ and

$$c_1 = \chi_l \text{ and } c_j = c_{j-1} + \frac{\chi}{M_{\text{out}} - 1}, \quad j = 2, \ldots, M_{\text{out}},$$

(3.2)

where $\chi = \chi_u - \chi_l$, where $\chi_l$ and $\chi_u$ are the lower and upper bounds of the output range. Now, recall from eq. (2.7) that $\tilde{C}_i$ is the only output MF parameter needed to compute the output of an FM. Hence, the type of output MF can be any MF, such as singleton, gbell or triangular. However, singleton MFs can be presented with only one parameter and therefore they are used in this thesis.

### 3.1.3 An Illustrative Initialization Example

In this Section the DT initialization method which was proposed in P3 is used to create the first FM of the initial population for a regression problem Mortgage which has 15 input variables and 1049 data points. This data set is available from KEEL-website\(^1\). Moreover, data partitions for five-fold cross-validation (5-CV) are also available from that website and this example first uses the training data set of the first data partition to illustrate the functioning of the proposed initialization method. After that, the procedure of 5-CV is performed and the obtained results are compared to frequently used WM initialization method.

First consider the training set of the first data partition which has 839 data points. In order to use the DT initialization method, user needs to provide the value of $M_{\text{out}}$, which determines the granularity of discretization. Usually, for this problem which has a moderate number of input variables, $M_{\text{out}}$ is either three or five [7], [P4]. The rule of thumb is that for low-dimensional problems, $M_{\text{out}}$ is higher than for high-dimensional problems. By increasing the value of $M_{\text{out}}$, the initial FMs usually become more complex but also more accurate.

In this example, the output data are discretized into three regions and the result is shown in Fig. 3.2. After that a DT is constructed and shown in Fig. 3.3. One can see that the resulting DT uses only five input variables, namely $x_1, x_5, x_6, x_9$.

\(^1\)http://sci2s.ugr.es/keel/datasets.php
and $x_{12}$. The rest ten input variables are discarded. This is the main benefit of DT initialization. DT can be presented as a set of rules which are created by following from the root of the tree to each of the leaves (i.e. the number of rules correspond to the number of leaves). Two example rules are given as:

Rule 1: If $x_6 > 7.27$ and $x_1 > 13.58$, then class is 3.

Rule 2: If $x_6 \leq 7.27$ and $x_1 > 6.24$ and $x_5 \leq 6.05$, then class is 1.

These rules can be presented with fuzzy sets instead of inequalities [3]. The resulting fuzzy partitions are shown in Fig. 3.3 and it can be seen that the MFs intersect at those points which were used in the inequalities of the DT. It is also noticed that the fuzzy partition of $x_1$ is not transparent but contains highly overlapping MFs. This happens because $x_1$ is present on both sides of the DT. Hence, it is difficult to assign meaningful linguistic values for this input variable. The fuzzy partitions of other input variables are transparent and the output fuzzy partition is created according to eq. (3.2) and presented here with gbell MFs instead of singleton MFs for illustrative purposes.
Figure 3.3: Top: Obtained DT for Mortgage problem. Bottom: The fuzzy partitions obtained by transforming DT into an FM. $x_1$ is partitioned with 4 MFs, which overlap each other severely. Other input variables are partitioned with 2 MFs and they do not overlap too much. Output is partitioned with 3 equally distributed uniformly shaped MFs for illustrative purposes.
Table 3.1: Averaged initialization results for Mortgage problem.

<table>
<thead>
<tr>
<th>Method</th>
<th>$R$</th>
<th>$R_{\text{cond}}$</th>
<th>MFs</th>
<th>$M_{\text{out}}$</th>
<th>$F$</th>
<th>$\text{MSE}_{\text{trn}}$</th>
<th>$\text{MSE}_{\text{tst}}$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>10.8</td>
<td>34.2</td>
<td>19.2</td>
<td>3</td>
<td>6.2</td>
<td>1.84</td>
<td>1.81</td>
<td>138.36</td>
</tr>
<tr>
<td>WM</td>
<td>75.8</td>
<td>1137</td>
<td>45.0</td>
<td>3</td>
<td>15.0</td>
<td>1.72</td>
<td>1.65</td>
<td>1350.8</td>
</tr>
</tbody>
</table>

**Five-Fold Cross-Validation**

In order to compare results reliably with WM initialization method, 5-CV procedure was run for both methods. This means that training was performed with four subsets of data and testing with the remaining subset. This is repeated five times, such that, testing data set is changed to another subset every time; the subset which was previously used for testing is now used for training and one of the four subsets which was used for training becomes the testing set.

WM initialization method requires that user provides not only the value of $M_{\text{out}}$ but also the value of $M_{\text{in}}$ which determines the number of input MFs. Here, both $M_{\text{in}}$ and $M_{\text{out}}$ were set to three and the fuzzy partitions consist of uniformly shaped evenly distributed MFs which correspond to the MFs of the output fuzzy partition in Fig. 3.3.

Table 3.1 shows the averaged results of 5-CV. It can be seen that DT initialization leads to less complex FMs which have less rules $R$, rule conditions $R_{\text{cond}}$, input MFs (denoted by MFs), and input variables $F$ than the FMs initialized by WM method. It must be kept in mind, however, that all fuzzy partitions of the FMs obtained by WM method are transparent, whereas one fuzzy partition obtained by DT method is not transparent.

Accuracy is measured by mean squared error divided by two\(^2\):\(^2\)

\(^2\)It is quite common to denote mean squared error divided by two simply by MSE in the field of MGFSs even though it may be confused to mean squared error.
\[ \text{MSE} = \frac{1}{2D} \sum_{k=1}^{D} (y_k - \hat{y}_k)^2, \]  

(3.3)

where \( D \) is the number of data points and \( y_k \) and \( \hat{y}_k \) are the actual and predicted output, respectively. MSE for training and test sets is denoted as \( \text{MSE}_{\text{trn}} \) and \( \text{MSE}_{\text{tst}} \), respectively. It can be seen that WM method leads to slightly better accuracy in this case. The accuracy, however, can be significantly improved by MOEA optimization. For example, in [7], \[P4\], \( \text{MSE}_{\text{tst}} < 0.10 \) was reported.

Since the purpose of initialization is to ease the further optimization by MOEA, it is interesting to see the effect of the initialization methods on the number of parameters to be optimized. When both rule base and data base are optimized and singleton output MFs are used, the number of parameters to be optimized is \( \theta = R(F+1) + \rho \times \text{MFs} + M_{\text{out}} \) (see also Section 2.5.2), where \( \rho = 3 \) because \( \text{gbell} \) input MFs are used throughout this thesis. Hence, for WM method \( \theta = 75.8 \times (15+1) + 3 \times 45 + 3 = 1350.8 \) and for DT method \( \theta = 10.8 \times (6.2+1) + 3 \times 19.2 + 3 = 138.36 \).

To summarize, the complexity of initial FMs and the number of parameters to be optimized by MOEA is significantly reduced by DT initialization. There is, however, room for improvement with regards to transparency of input fuzzy partitions. The MGFSs in Chapter 4 propose a novel initialization method which overcomes this problem by combining the benefits of DT and WM initialization methods.

### 3.2 Further Optimization by NSGA-II

In all publications of this thesis, NSGA-II was used to further optimize the initial population. The original implementation of NSGA-II supports problems which have binary and/or real-coded variables. In this thesis, the rule base is presented by integers and MF parameters by real-coded variables. Thus, NSGA-II is not suitable as it is for solving this kind of problems appropriately but the genetic operators should be replaced with genetic operators suitable for handling integers appropriately (see also Section 2.5.3).
In the first MGFS proposals (P1–P3) the genetic operators of NSGA-II were used as they are. The aforementioned problem was circumvented by using real-coding for all variables and rounding the value to nearest integer if the variable was an integer-coded variable. As it was shown in Section 2.5.3, this is not effective way to solve the problem. Thus, in the second MGFS proposals (P4 and P5) in Chapter 4, the genetic operators of NSGA-II were replaced with novel genetic operators. These genetic operators enabled optimization of integer and real-coded variables without any rounding operators.

The rest of this Section describes the components of further optimization in the first MGFS proposals. First, the coding scheme for representing the FMs, such that NSGA-II can be used, is described. Then, the objectives functions are discussed. Finally, genetic operators are presented in Section 3.2.3.

### 3.2.1 Coding of FMs

In this thesis Pittsburgh style EA [84] is considered. Therefore, each chromosome is an FM, however, only rule base and MF parameters of FM are subject to optimization. The antecedent part of the rule base is denoted as \( A \). Although \( A \) actually contains real-coded variables, it is presented here with integers, which are obtained by rounding the real-coded variables to nearest integers. That is done in order to simplify the notation. The first FM of the initial population (denoted as initial FM) determines the maximum number of rules, input variables and MFs per input and output variable in all other FMs. If that FM has \( R \) rules and \( n_s \) input variables, \( A \) is given as:

\[
A = \begin{bmatrix}
    a_{1,1} & a_{1,2} & \cdots & a_{1,n_s} \\
    a_{2,1} & a_{2,2} & \cdots & a_{2,n_s} \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{R,1} & a_{R,2} & \cdots & a_{R,n_s}
\end{bmatrix},
\]  

(3.4)
where \( n_s \) is the number of input variables selected from \( n \) input variables during the initialization. Of course, \( n_s \leq n \), but usually \( n_s < n \). Each \( a_{i,j} \in \{0, 1, \ldots, M_j\} \) indicates which MF is used for variable \( j \) in rule \( i \). \( M_j \) is the number of MFs in input variable \( j \) in initial FM.

This coding scheme is flexible in the sense that it allows the number of rules, rule conditions and input variables to be altered during the further optimization. If input variable \( j \) is not used in rule \( i \), then \( a_{i,j} = 0 \). If rule \( i \) is not used in an FM, then \( \forall j, a_{i,j} = 0 \). If input variable \( j \) is not used in an FM at all, then \( \forall i, a_{i,j} = 0 \).

Input MF parameters are contained in a real-coded matrix \( P \):

\[
P = \begin{bmatrix}
p_{1,1} & p_{1,2} & \cdots & p_{1,\delta} \\
p_{2,1} & p_{2,2} & \cdots & p_{2,\delta} \\
\vdots & \vdots & \ddots & \vdots \\
p_{\rho,1} & p_{\rho,2} & \cdots & p_{\rho,\delta}
\end{bmatrix},
\]

where \( \rho \) is the number of parameters used to define an MF and \( \delta = \sum_{j=1}^{n_s} M_j \) is the total number of MFs in initial FM.

In P1 and P2, consequent part of the fuzzy rule was not included into chromosome. It was static and created in initialization phase. However, in P3 consequent part was also included into chromosome. That was done, because during the further optimization, the antecedents of the rules and MF parameters are modified, which may lead to a situation in which the initial rule consequents are not adequate. Thus, in P3, real-coded vector \( s = [s_1, s_2, \ldots, s_R]^T \), which states the output MFs of the rules, was also included into chromosome. Similarly to \( A \), \( s \) also needs to be rounded to nearest integer when fitness evaluation is performed. Each \( s_i \in \{1, \ldots, M_{\text{out}}\} \) indicates which output MF is used in rule \( i \). The output MF parameters were static and assigned in initialization phase (see also Section 3.1.2).

To summarize, in P1 and P2 which dealt with classification problems, NSGA-II was used to optimize rule antecedents and input MF parameters for the class
labels specified in initialization phase. The total number of parameters $\theta_{P1,P2}$ to be optimized was therefore:

$$\theta_{P1,P2} = Rn_s + \rho \delta.$$  \hspace{1cm} (3.6)

In P3 which dealt with regression problems, rule consequents were also optimized and therefore the total number of parameters $\theta_{P3}$ to be optimized was:

$$\theta_{P3} = Rn_s + \rho \delta + R.$$  \hspace{1cm} (3.7)

### 3.2.2 Objective Functions

Measuring accuracy is easier than measuring interpretability which is a subjective matter [46]. For classification problems, accuracy can be measured, for example, by misclassification rate, area under receiver operating characteristics (AUC) [76], true positive (TP) and false positive (FP) rates or by Cohen’s kappa [22]. Most common of these measurements is the misclassification rate, which is computed by dividing the number of data points which were incorrectly classified by the total number of data points in the data set. Misclassification rate was used in P1 (see also Table 3.2). It has been, however, criticized that misclassification rate is not a good metric when class distributions and misclassification costs are not known [77], [36], [83], [14]. In P2 dealing with binary classification problem with highly imbalanced class distribution, TP and FP rates were used instead:

$$\text{TP rate} = \frac{\text{positives correctly classified}}{\text{total positives}},$$  \hspace{1cm} (3.8)

$$\text{FP rate} = \frac{\text{negatives incorrectly classified}}{\text{total negatives}},$$  \hspace{1cm} (3.9)

where TP rate and FP rate $\in [0, 1]$. In P2 the purpose was to raise an alarm whenever harmful bioaerosols were in the air and to minimize the false alarms. There-
Table 3.2: The objective functions used in publications.

<table>
<thead>
<tr>
<th>Publication</th>
<th>Problem type</th>
<th>Objective 1</th>
<th>Objective 2</th>
<th>Objective 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Classification</td>
<td>Misclassification rate</td>
<td>$R$</td>
<td>$R_{\text{cond}}$</td>
</tr>
<tr>
<td>P2</td>
<td>Classification</td>
<td>TP rate</td>
<td>FP rate</td>
<td>$T$</td>
</tr>
<tr>
<td>P3</td>
<td>Regression</td>
<td>Mean squared error</td>
<td>$R$</td>
<td>$R_{\text{cond}}$</td>
</tr>
</tbody>
</table>

Therefore, the positives and negatives were the data points presenting harmful and safe bioaerosols, respectively. When regression problems are considered, it is common to use either mean squared error or root mean squared error. Mean squared error was applied in P3 and it is computed by multiplying MSE in eq. (3.3) by two.

Interpretability of FMs was measured in P1, P3 quantitatively by computing the number of rules $R$ and rule conditions (total rule length) $R_{\text{cond}}$. The purpose was to minimize $R$ and $R_{\text{cond}}$ in order to improve the interpretability. In P2 no quantitative interpretability measures were optimized. Instead, a qualitative measure, namely transparency index $T$, was used. It measures the transparency of fuzzy partitions and is minimized ($T = 0$) when MFs are distinguishable and UoD is strongly covered (see Fig. 3.4(d)). That transparency index was extended from Kim et al. [62] by introducing a term which reduces the effect of relaxed covering [89]. More specifically, $T$ consists of three penalties which are proportional to the severity of transparency deterioration. The formulas for computing these penalties are given in P2 and here they are discussed in general terms:

- An overlap penalty $P_{\text{OL}}$ is issued if the membership value at intersection point of two neighboring MFs is larger than $\alpha_H$ (Fig. 3.4(a)). $\alpha_H$ was set to 0.6 in the experiments.

- A discontinuity penalty $P_{\text{DC}}$ is given if the membership value at the intersection point of two neighboring MFs less than $\alpha_L$ (Fig. 3.4(b)). $\alpha_L$ was set to 0.1 in the experiments.
• A middle value penalty $P_{MV}$ is issued if at the center of an MF another MF receives membership value larger than $\alpha_L$ (Fig. 3.4(c)).

### 3.2.3 Genetic Operators

Genetic operators of NSGA-II were used in P1–P3 without any modifications. Thus, polynomial mutation and simulated binary crossover were the genetic operators. These standard operators are not discussed in this thesis and the reader is referred to [29] for more information. They alter the values of $A$, $P$, that is, the rule antecedents and input MF parameters. In P3, also the rule consequents, $s$, were altered. Each variable was constrained between lower and upper limits, which remained the same during the further optimization (i.e. static constraints). Hence, in P1 and P3 transparency was not guaranteed. In P2 transparency of fuzzy partitions depended on the value of transparency index $T$. There existed FMs with very transparent fuzzy partitions (i.e. the value of $T$ was low) and FMs with not very transparent fuzzy partitions (i.e. the value of $T$ was high).
As an additional operator to the standard genetic operators of NSGA-II, P2 and P3 used a heuristic rule removal operator to remove inconsistent rules and rule conditions. It was pointed out in P2 that these operators reduce computational costs because the complexity of the rule base is reduced and therefore the fitness evaluations are computationally less demanding.

3.3 Summary of the First MGFSs of This Thesis

The first MGFS proposals in P1–P3 can usually obtain compact and accurate FMs, however, there is still room for improvement with regards to transparency of fuzzy partitions, initialization and genetic operators. Transparency of fuzzy partitions was taken into account only in P2 and not necessarily the best way. Through the experiments in P2, it was confirmed that by including a transparency index $T$ as an objective function, transparency of fuzzy partitions was improved on average. Furthermore, by minimizing $T$, usually complexity of FMs was reduced as well. It was, however, noticed that the values of $T$ in population may vary significantly; some of the FMs can have highly transparent fuzzy partitions whereas some others can have unacceptable fuzzy partitions. The variation interval can be constrained in order to reduce the variation. However, because the genetic operators were not specially designed to obtain offspring with transparent fuzzy partitions, many FMs of the offspring population usually had unacceptable value of $T$. In other words, many of the offspring were infeasible, which deteriorated the search efficiency.

The second MGFS proposals in next Chapter will tackle this problem by using dynamic constraints in order to guarantee transparency of fuzzy partitions. User will set the transparency level he/she desires and all FMs will always meet these requirements in each input variable. A novel initialization method which guarantees transparency of fuzzy partitions will be proposed as well. Finally, novel genetic operators which work together with the dynamic constraints will be proposed.
Chapter 4

Proposed Multiobjective Genetic Fuzzy Systems with Dynamic Constraints (P4 and P5)

The aim of P4 and P5 was to design multiobjective genetic fuzzy systems (MGFSs) which are capable of guaranteeing transparency of fuzzy partitions of each fuzzy model (FM) to a user-defined level. In order to do that, the standard genetic operators of the nondominated sorting genetic algorithm II (NSGA-II) were replaced with novel genetic operators. Membership functions (MFs) were tuned with a novel dynamic tuning strategy, which requires that before and after of each alteration of MF parameters, transparency of fuzzy partitions must meet certain transparency criteria. Therefore, a new initialization method, which guarantees transparency of fuzzy partitions in initial population was also proposed. Therefore, the orientation in these second MGFS proposals is quite different from the first MGFS proposals in Chapter 3. It is summarized in Fig. 4.1.

This new orientation has some benefits. 1) User can define the transparency level he/she wants. All obtained FMs will meet those needs. 2) Because transparency index is not an object to be minimized, there is one objective less to be minimized. This is beneficial because many of the current multiobjective evolutionary algorithms (MOEAs), including NSGA-II, do not perform well on problems with many objectives, where many is usually four or more [44,53]. 3) The proposed ge-
Create an initial population of FMs which have transparent fuzzy partitions

Optimize the initial population by MOEA:
- Perform rule learning, granularity learning, input variable selection and dynamic MFs tuning

Result:
A set of Pareto optimal FMs which have transparent fuzzy partitions

Figure 4.1: Schemata of the second MGFS proposals to identify FMs based on data. All FMs in all phases of optimization have transparent fuzzy partitions.

Genetic operators are tailored, such that, both real-coded and integer-coded variables are handled appropriately.

The rest of this Chapter is organized as follows. Since each FM has to fulfill certain transparency conditions, they are introduced first. Dynamic input MFs tuning strategy, which guarantees that these transparency conditions are fulfilled when MFs are tuned, is presented after that. Transparency of output fuzzy partitions is maintained by static constraints, since only one parameter is used to define an output MF (Section 4.3). Since the dynamic tuning strategy requires that initially fuzzy partitions must be transparent, simple partitioning algorithms are introduced in Section 4.4. They are used during the initialization (Section 4.5) and also as a part of genetic operators during the further optimization (Section 4.6). Finally, Section 4.7 summarizes this Chapter.

4.1 Fuzzy Partition Transparency Conditions

Like mentioned in Section 1, globally defined MFs are used in this thesis, which means that the same MF definitions are used in all rules. In P4 and P5, it is considered that a fuzzy partition is transparent if the frequently used transparency conditions by de Oliveira [28] are met:
(a) The number of MFs per variable is moderate.

(b) MFs are distinguishable, that is, two MFs do not present the same or almost the same linguistic meaning.

(c) Each MF is normal. An MF is normal if it has membership value 1 at least at one point of universe of discourse (UoD).

(d) UoD is strongly covered. At least one MF receives a membership value $\beta$ (where $\beta > 0$) at any point of UoD.

It can be seen that the conditions (a) and (c) are additional requirements to the requirements in Section 3.2.2. Moreover, unlike in Section 3.2.2 where the aim was to satisfy the transparency requirements as well as possible, here these requirements must be met by all FMs at any given phase of optimization.

Condition (a) is satisfied simply by constraining the maximum number of MFs to a moderate number (e.g. 3, 5, 7, or 9). The normality condition (c) is also easily satisfied by applying normal MFs, such as gbell MFs, and by applying genetic operators which do alter their normality. Satisfying the distinguishability condition (b) and the coverage condition (d) is, however, not trivial. Common attempt to satisfy them is the application of static constraints, which allow the MF parameters to be modified from their initial values only to a certain extent. However, like discussed in Section 1.1, this approach is suboptimal. Instead, the constraints should be updated every time MF parameters are modified. In other words, the constraints should be dynamic. Therefore, in P4 a dynamic tuning strategy was proposed. It requires that before and after of each alteration of MF parameters, the following four conditions must be satisfied:

1. **Symmetry condition**: All MFs are symmetrical. Gbell MFs, which are used in this thesis, are symmetrical by definition.

2. **$\alpha$-condition**: At any intersection point of two MFs, the membership value is at most $\alpha$.

3. **$\gamma$-condition**: At the center of each MF, no other MF receives membership value larger than $\gamma$. For gbell MFs, center is the parameter $c$. 
Figure 4.2: (a) An example fuzzy partition which violates $\alpha$, $\beta$, and $\gamma$-conditions when $\beta = 0.05$, $\gamma = 0.25$, and $\alpha = 0.8$. (b) An example fuzzy partition which meets the transparency conditions. The MF centers are shown with dotted vertical lines.

4. $\beta$-condition: UoD is strongly covered, that is, at each point of UoD at least one MF has membership value at least $\beta$.

From these conditions only the first condition is easily satisfied. The rest three conditions require updating the corresponding dynamic constraint before each alteration of MF parameters. This means, for example, that if the width of a gbell MF (i.e. parameter $a$) is increased, the maximum width depends on the parameters of its neighboring MF(s), such that the aforementioned conditions are not violated. The formulas for updating the constraints are documented in full detail in P4 but since they are rather lengthy, it is not meaningful to show them here.

Instead of showing the formulas, Section 4.2 shows an example of dynamic MFs tuning. Here, Fig. 4.2(a) shows an example fuzzy partition which violates $\alpha$, $\beta$, and $\gamma$-conditions when $\beta = 0.05$, $\gamma = 0.25$, and $\alpha = 0.8$. In contrast to P2, in
which a penalty would be issued to such a partition, these second MGFS proposals in P4 and in P5 do not allow this kind of fuzzy partitions at all. Instead, all fuzzy partitions must always meet the transparency conditions. An example of such fuzzy partition is shown in Fig. 4.2(b). In this case, for example the width of each of the three MFs could be still increased without violating the transparency conditions.

Selection of $\alpha$, $\beta$, and $\gamma$ depends on the initialization algorithm that is being used and of the desired level of transparency. The initialization algorithms in this thesis guarantee that at any intersection point of two MFs, the membership value is 0.5. Moreover, partitions with only one MF are not allowed and at extreme points of UoD the membership value is 1. Finally, at the center of each MF, none of the other MFs can receive a membership value larger than $\gamma^* = 0.05$. Therefore, $0 < \beta < 0.5$, $\gamma^* < \gamma < 0.5$, and $0.5 < \alpha < 1$ must be selected in order to apply the dynamic MFs tuning strategy. P4 and P5 use $\beta = 0.05$, $\gamma = 0.25$, and $\alpha = 0.8$. Moreover, P4 uses also $\beta = 0.4$, $\gamma = 0.1$, and $\alpha = 0.6$ in order to obtain higher level of transparency. It was, however, pointed out in P4 that there exists a trade-off between transparency of fuzzy partitions and accuracy.

### 4.2 Dynamic Input MFs Tuning

In this Section an example of modifying the gbell MF parameters $a$, $b$, and $c$ is given. The modification is performed by the proposed genetic operators introduced later in Section 4.6.3. It is important to emphasize that before and after each alteration, the fuzzy partition must meet the transparency conditions in Section 4.1.

The initial fuzzy partition of this example is illustrated in Fig. 4.3 (a). If that partition is modified by decreasing the width (i.e. parameter $a$) of MF 2, the lower limit of $a$ is computed depending on the parameters of its neighboring MFs 1 and 3. Fig. 4.3 (b) shows the result when the minimum value for $a$ is assigned. If $a$ were to be decreased more, the $\beta$-condition would be violated.

Now, starting again from Fig. 4.3 (a) but instead of modifying the parameter $a$, parameter $b$ defining MF’s shape is modified here. Fig. 4.3 (c) shows the result
when minimum value for $b$ is assigned. If $b$ were to be decreased more, the $\gamma$-condition would be violated.

Finally, the starting point is again Fig. 4.3 (a) but the modifiable parameter is $c$ which defines the center of an MF. Fig. 4.3 (d) shows the resulting fuzzy partition when the maximum value for $c$ is assigned. In this case MF 2 cannot be moved towards right anymore without violating the $\alpha$-condition. However, if the parameters of neighboring MF 3 are modified, then it may be possible to move MF 2 still towards right.

In this example the initial fuzzy partition was always Fig. 4.3 (a) and $a$, $b$, or $c$ of MF 2 was modified one at the time. It must be emphasized that it is not
necessary to start from the same partition or to modify only the parameters of MF 2. Instead, the proposed dynamic MFs tuning strategy is flexible in the sense that alteration operators can be performed one after another to any MF. For example, first parameter \(a\) of MF 4 is modified. Then, starting from the resulting partition, parameter \(c\) of MF 4 could be modified next. After that, starting from the resulting partition, parameter \(b\) of MF 5 could be modified. Moreover, in this example minimum or maximum values of \(a\), \(b\), or \(c\) were assigned for illustrative purposes. However, any value between the current parameter value and its current minimum or maximum value can be, of course, assigned.

So, by this tuning scheme the fuzzy partitions will always meet the user-specified requirements (specified by \(\alpha\), \(\beta\), \(\gamma\)) in case that the number of MFs remains the same. However, it is important to also learn the granularity of fuzzy partitions. Hence, the proposed genetic operators in Section 4.6.3 include some operators for altering the number of MFs as well. When the number of MFs is increased or decreased, the parameters of the existing MFs need to be modified somehow. It is not possible to add or remove MFs and to guarantee the transparency of fuzzy partitions without modifying the parameters of existing MFs. For example, if one or more MFs are removed, the remaining MFs need to modified, such that, there will not be any gaps in the fuzzy partition. Likewise, if the number of MFs is increased, the existing MFs need to be modified, such that, they do no overlap with the added MF(s).

In this thesis, this done as follows. First, the number of active MFs \(M_A\) in input variable after the alteration is determined. If \(M_A \geq 2\), a transparent fuzzy partition with \(M_A\) MFs is created. That fuzzy partition consists of unequally distributed MFs which are created by the simple algorithm in Section 4.4. If, however, \(M_A < 2\), then that input variable is removed. This input variable, however, can become active again, if at some point of optimization \(M_A\) for that variable is at least 2.

### 4.3 Static Output MFs Tuning

In case of regression problems, also the lateral displacements of output MFs are altered by the genetic operators. Since there is only one parameter to be modified, static constraints are suitable to constrain the locations of output MFs, such that,
the output MFs are allowed to move slightly left or right from their initial positions. This tuning strategy resembles lateral tuning method [5] and guarantees transparency of output fuzzy partition to a good level. Application of dynamic constraints might improve the accuracy slightly, but it was left for future study.

4.4 Partition Algorithms for Creating Transparent Fuzzy Partitions

P4 proposed simple partition algorithms for creating evenly and unevenly distributed transparent fuzzy partitions. These algorithms are necessary because the dynamic MFs tuning requires that initially the fuzzy partitions are transparent. Evenly distributed fuzzy partitions are used in the first FM of the initial population and an example of such partition with five MFs is shown in Fig. 4.4 (a).

In some cases, however, evenly distributed fuzzy partitions can be misleading if they do not present the real distribution of the data. Hence, it is beneficial if some MFs are, for instance, wide whereas some others are narrow. Another simple partition algorithm is proposed for that purpose and an example partition with five MFs is given in Fig. 4.4 (b). This algorithm is used in initialization phase to create the partitions of all FMs except the first FM for which evenly distributed MFs are used. Moreover, this algorithm is used during the further optimization in case that the number of MFs is altered by genetic operators.

The formulas for computing the MF parameters by these algorithms are documented in P4. It should be also noted that the algorithm for creating unevenly distributed MFs can also be used to create evenly distributed MFs, so actually only one algorithm is needed.

4.5 Initialization Methods

In order to use the dynamic MFs tuning strategy, the initialization methods need to be designed, such that, the transparency conditions in Section 4.1 are fulfilled by
each FM of initial population. Hence, the initialization methods of the first MGFS proposals (Chapter 3) of this thesis cannot be used without any modifications. In one sense, WM method or random initialization could be used as they can easily meet the transparency conditions. However, due to their drawbacks (Section 2.5.1), they were also excluded.

Instead, a new initialization method combining the benefits of WM and DT initialization methods was proposed in P4. In a nutshell, the input variable selection capability of DT is used to reduce the number of input variables, however, since the fuzzy partitions created by DT initialization may not be transparent, a simple partition algorithm, introduced in Section 4.4, is used to create evenly distributed fuzzy partitions. For those partitions, WM method is used to create rules. Since WM method may create extensive amount of rules for some data sets, this method is slightly modified in order to reduce the number of rules and rule conditions.
Figure 4.5: The new initialization method combining the benefits of WM and DT initialization methods. For classification problems, step 1 is skipped.

4.5.1 Initialization Method for Regression Problems

This initialization method is depicted in Fig. 4.5. The first two steps of this method, namely output data discretization and DT initialization, are identical to the initialization method presented in Fig. 3.1. Steps 3-5 are additional steps and explained next.

Unlike the earlier initialization methods in Section 3.1 which use the fuzzy partitions created by DT initialization, here the fuzzy partitions are created by the partition algorithm introduced in Section 4.4. In order to do that user must pro-
Proposed MGFSs with Dynamic Constraints (P4 and P5) 61

dive the number of input MFs \(M_{in}\) and the number of output MFs \(M_{out}\). This is a rather common practice and, for example, WM initialization and random initialization require that user provides these values. On the other hand, the initialization methods of the first MGFS proposals only require that user specifies \(M_{out}\) and DT determines \(M_{in}\) which, however, leads to high number of input MFs in some cases.

Even though user is required to specify \(M_{in}\), DT initialization can still be used to limit the number of input MFs. First, the number of MFs for each input variable \(j\) in the FM created by DT is checked and denoted by \(M_{DT,j}\). Then, instead of partitioning each input variable with \(M_{in}\) MFs, each input partition is created with \(\min(M_{DT,j}, M_{in})\) MFs. Finally, the output is partitioned with \(M_{out}\) MFs. So, this approach utilizes DT to determine the number of MFs only when it assigns less than \(M_{in}\) MFs to some input variables. Otherwise, the number of MFs in input variable is \(M_{in}\).

The rule base created by DT initialization is also discarded. That is done because: 1) for some problems the number of rules and rule conditions can be high and 2) because DT initialization created the rules using the MFs obtained by transforming the DT into an FM. Now, because those MFs are no longer used, it is not meaningful to use those rules either.

Instead, rule base is created by WM algorithm with some novel modifications. 1) when a data point is matched to MFs in order to generate a rule, the data point is not always matched to MFs of all possible input variables. Instead, it is first classified by the constructed DT and only those input variables, which were used by DT to classify the data point are used for matching. 2) As WM algorithm may create large number of rules for data sets with many data points and/or input variables, the generated rules are divided among the members of initial population and at most \(R_{max} = 30\) of them are included into one FM.

Once the rule base is created, the number of active MFs \(M_{A,j}\) for each input variable \(j\) is checked (an MF is active if it is part of at least one of the rules). Those input variables which have less MFs than initially purposed, (i.e. \(M_{A,j} < \min(M_{DT,j}, M_{in})\)) have gap in the fuzzy partition and the whole UoD is not strongly covered. Depending whether \(M_{A,j} \geq 2\) or \(M_{A,j} < 2\), either a new evenly distributed partition is created or the variable is removed, respectively. That is done in order to guarantee that all FMs in initial population have transparent fuzzy partitions.
4.5.2 An Illustrative Initialization Example

In order to understand the functioning of this new initialization method better, let us return to the illustrative example from Section 3.1.3. So, the problem at hand is Mortgage data set with 1049 data points and 15 input variables. In Section 3.1.3 the proposed DT initialization method led to quite compact FMs with reasonable accuracy. However, the fuzzy partition of one input variable was not transparent. On the other hand, WM initialization method led to significantly more complex rule base but transparency of all the fuzzy partitions was high.

Like mentioned previously, the first two steps of the new initialization method are exactly the same as in the initialization method in Section 3.1.2. So, first exactly the same fuzzy partitions as in Fig. 3.3 are obtained. Then $M_{DT_j}$ for the selected input variables $j \in \{1, 5, 6, 9, 12\}$ is determined. It can be seen from Fig. 3.3 that $M_{DT_1} = 4, M_{DT_5} = 2, M_{DT_6} = 2, M_{DT_9} = 2$, and $M_{DT_{12}} = 2$. For each input variable the number of input MFs is determined by $\min(M_{DT_j}, M_{in})$, where $M_{in} = 3$ (see Section 3.1.3). So, for $x_1$ three input MFs are assigned, whereas two input MFs are assigned for the other input variables. Output is partitioned with $M_{out} = 3$ MFs. The fuzzy partitions are created by the partition algorithm in Section 4.4 and consist of uniformly shaped equally distributed MFs.

After that, rules are created for those fuzzy partitions using the WM algorithm aided by DT. So, each data point of the training set is first classified with the DT and the input variables which were used to classify the data point are determined. For example, consider the first data point of the training set. Its value for input variables $x_1, x_5, x_6, x_9$, and $x_{12}$ are respectively 6.66, 5.74, 7.63, 198.6, 759.2 and the value for output is 6.77 which, when discretized, corresponds to class 2 like seen from Fig. 3.2. Now, classify this data point by the DT in Fig. 3.3. At the node of the tree there is a test $x_6 > 7.27$ and because $x_6$ is 7.63, the left branch of the tree is selected. The next test is $x_1 > 13.58$ but because $x_1 = 6.66$, the right branch is selected now. Finally, the next test is $x_{12} > 828.4$ and again right branch is selected because $x_{12} = 759.2$. This completes the classification of this data point, since the leaf with class value 2 is reached. So, in this case DT classified the data point correctly and used the input variables $x_1, x_6$, and $x_{12}$.

Then, for each $x_1, x_6$, and $x_{12}$ separately, the data point is matched to the MFs
which were created in step 3 of Fig. 4.5. The MF which matches the data point to highest degree becomes the rule condition for that rule. In this example the MFs for $x_1$, $x_6$, and $x_{12}$ are exactly the same as in Fig. 4.6, which presents the MFs after the whole initialization procedure is completed. However, this is not always the case because the fuzzy partitions may still change when the activity of MFs is checked in step 5 of Fig. 4.5. Similarly, output is matched to the output MFs\footnote{This step can also be completed by simply assigning the consequent of the rule to the output MF corresponding the discretized output value, which in this case is class 2.} and the MF with the highest match becomes the consequent of the rule. This procedure is repeated for each data point of the training set. Finally, duplicate rules are removed.

Then, the number of active MFs $M_{A_j}$ for each input variable is determined. All MFs which received the highest match during the rule generation phase for at least one data point are active. It is noticed that $M_{A_1} = 3$, $M_{A_5} = 1$, $M_{A_6} = 2$, $M_{A_9} = 2$, and $M_{A_{12}} = 2$. Hence, it is noticed that $M_{A_j} = \min(M_{DT_j}, M_{in})$ for $j \in \{1, 6, 9, 12\}$.\footnote{This step can also be completed by simply assigning the consequent of the rule to the output MF corresponding the discretized output value, which in this case is class 2.}

**Figure 4.6:** The fuzzy partitions for Mortgage problem with the new initialization method. All partitions are transparent. Output is partitioned with gbell MFs for illustrative purposes.
Table 4.1: Averaged initialization results for Mortgage problem.

<table>
<thead>
<tr>
<th>Method</th>
<th>$R$</th>
<th>$R_{\text{cond}}$</th>
<th>MFs</th>
<th>$M_{\text{out}}$</th>
<th>$F$</th>
<th>$\text{MSE}_{\text{trn}}$</th>
<th>$\text{MSE}_{\text{tst}}$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>10.8</td>
<td>34.2</td>
<td>19.2</td>
<td>3</td>
<td>6.2</td>
<td>1.84</td>
<td>1.81</td>
<td>138.36</td>
</tr>
<tr>
<td>WM</td>
<td>75.8</td>
<td>1137</td>
<td>45.0</td>
<td>3</td>
<td>15.0</td>
<td>1.72</td>
<td>1.65</td>
<td>1350.8</td>
</tr>
<tr>
<td>DT + WM</td>
<td>19.4</td>
<td>55.0</td>
<td>10.6</td>
<td>3</td>
<td>4.4</td>
<td>0.737</td>
<td>0.449</td>
<td>139.56</td>
</tr>
</tbody>
</table>

For $x_5$, there is only one active MF and that input variable is therefore removed. The fuzzy partitions for the rest of the variables are shown in Fig. 4.6 and it can be seen that they are highly transparent. The rule base consists of 12 rules and 33 rule conditions. Since there are no more than $R_{\text{max}} = 30$ rules, all rules are included into the rule base and the initialization procedure is completed.

In order to compare reliably the obtained results to the previous results in Section 3.1.3, the five-fold cross-validation (5-CV) was performed. The averaged results of 5-CV are shown in Table 4.1 and this new initialization method is denoted as DT + WM. It is seen that the FMs obtained by DT + WM have less rules $R$, rule conditions $R_{\text{cond}}$, input MFs (MFs), and input variables $F$ than the FMs obtained by WM. It is also seen that the FMs obtained by DT + WM have more rules and rule conditions but less input MFs and input variables than the FMs obtained by DT. DT + WM is the most accurate in both training and test sets. Finally, the number of parameters to be optimized, $\theta$, is practically the same as with the DT method and significantly lower when compared to WM method.

So, it can be seen that this new initialization method (DT + WM) overcomes the shortcomings of DT and WM initialization methods. It obtains quite compact FMs with highly transparent fuzzy partitions with flexible amount of MFs. Moreover, the FMs are reasonably accurate for initial FMs as well. Finally, the parameters to be optimized is dramatically reduced when compared to WM method.
4.5.3 Initialization Method for Classification Problems

Only slight modifications are needed to make the initialization method presented in Section 4.5.1 to be used also in classification problems. 1) Output data discretization is not needed. 2) When the slightly modified WM algorithm is used, only the input data is matched to input MFs and the consequent of the rule is simply the class label of the given data point.

4.6 Further Optimization by NSGA-II

After the population is initialized it must be coded, such that, it can be optimized by NSGA-II. This Section presents the chromosome coding followed by objective functions and genetic operators.

4.6.1 Coding of FMs

The antecedents of the rule base are presented with integers unlike in Section 3.2.1 in which real-coding was used. Besides that, there are no other differences in presenting the antecedents of the rule base. Parameters of the input MFs are presented the same way as in Section 3.2.1.

Consequents of the rule base are included into chromosome for both classification and regression problems. In Section 3.2.1, that was the case only for regression problems. The consequents are presented with integer-coding. Finally, when regression problems are considered, the output MF parameters are also included into chromosome. They are defined in a real-coded vector \( o = [o_1, o_2, \ldots, o_{M_{out}}]^T \) presenting the center of an output MF. Therefore, when regression problems are considered, total number of parameters \( \theta_{p4} \) to be optimized is:

\[
\theta_{p4} = Rn_s + \rho \delta + R + M_{out}. \tag{4.1}
\]
When classification problems are considered, total number of parameters $\theta_{P5}$ to be optimized equals to $\theta_{P3}$, which is defined by eq. (3.7).

### 4.6.2 Objective Functions

In both P4 and P5, two objective functions are subject to optimization. In P4 the aim is to 1) minimize mean squared error (MSE) divided by two (see eq. (3.3)) and to 2) minimize the number of rule conditions $R_{\text{cond}}$. In P5 the second objective is the same as in P4, but the first objective is misclassification rate.

### 4.6.3 Genetic Operators

Like mentioned previously, the motivation to replace the original genetic operators of NSGA-II was to enable integer-coding of rule base and to enable dynamic MFs tuning. Thus, in P4 five mutation and crossover operators were specially tailored to enable dynamic MFs tuning strategy together with rule learning (see Fig. 4.7). In P5 four of them are used. The one which is excluded is the operator which modifies the lateral displacements of output MFs. That operator is not needed because P5 deals with classification problems.

The proposed genetic operators are shown in Fig. 4.7. The reasoning for several genetic operators is that together they enable modification of rule antecedents, rule consequents, input and output MF parameters. Moreover, rule learning enables input variable selection and granularity learning because the rule base coding scheme, defined in Section 3.2.1, allows that. One can also see that the mutation and crossover operators 1, 4, and 5 are very similar to each other. The difference between them is that mutation operators use only one FM and modify its parameters (i.e. unary operator), whereas crossover operators select two FMs as parents (i.e. binary operator) and the parameters of the children depend on their parent’s parameters.

In different stages of optimization, some of these genetic operators may not be applicable. Thus, when a genetic operator is performed, one of the currently applicable operators is randomly selected by uniform chance. Probability for applying
Proposed MGFSs with Dynamic Constraints (P4 and P5)

Mutation Operators

**Operator 1:** Dynamic input MFs tuning

**Operator 2:** Creates new unevenly distributed fuzzy partition

**Operator 3:** Modifies rule antecents

**Operator 4:** Modifies a rule consequent

**Operator 5:** Static output MFs tuning

Crossover Operators

**Operator 1:** Dynamic input MFs tuning

**Operator 2:** Swaps rule conditions and MF parameters of an input variable

**Operator 3:** Swaps rule antecents

**Operator 4:** Swaps a rule consequent

**Operator 5:** Static output MFs tuning

Legend:

- MFs tuning
- Rule learning
- Rule learning and MFs tuning

**Figure 4.7:** Proposed genetic operators for MFs tuning and rule learning. Rule learning also enables input variable selection and granularity learning.

crossover is \( P_c = 0.1 + \frac{G}{G_{\text{Tot}}} \), where \( G \) is the current generation and \( G_{\text{Tot}} \) is the total number of generations. If crossover was applied, mutation is applied with probability \( P_m = 0.1 \) and if crossover was not applied mutation is always applied. This strategy is similar to strategy applied in [19]. The genetic operators are explained next in general level and more detailed descriptions are available in P4.

**Mutation Operators**

**Operator 1 – Dynamic MFs Tuning:** This operator modifies the input MF parameters individually. First, it selects one or more input variables for which this operator is performed. Those input variables must have at least two active MFs. Then, from each of these selected input variables, one MF and one of its parameters \((a, b, \text{ or } c)\) are selected. After that, these parameters are modified according to the dynamic MFs tuning strategy, such that, after the modification the transparency conditions are met. An example of dynamic MFs tuning was given in Section 4.2.
Operator 2 – A New Unevenly Distributed Fuzzy Partition: Instead of modifying the input MF parameters individually, like the previous operator, this operator modifies them more drastically by selecting an input variable for which $M_A \geq 2$. It then creates a new unevenly distributed partition with $M_A$ MFs using the algorithm presented in Section 4.4. So, this operator explores the search space quite drastically, whereas the previous operator is closer to fine tuning. These operators complement each other and allow exploration and local search of fuzzy partitions.

Operator 3 – Rule Antecedents Learning: This operator modifies $n_{\text{rule cond}}$ randomly selected rule conditions $a_{i,j}$ (see eq. (3.4)), where $n_{\text{rule cond}} \in [1,10]$ is a random integer. The selected rule conditions are replaced with random rule conditions, however, as it is easier to obtain compact than accurate FMs [37], this operator favors non-zero replacement conditions during the first half of the total number of generations $G_{\text{Tot}}$.

This operator may alter the number of MFs in input variable and therefore enables granularity learning. It may also alter the number of input variables, since the coding scheme in Section 3.2.1 allows that as well. It was mentioned in Section 4.2 that when the number of MFs in input variable is increased or decreased, the parameters of existing MFs need to be modified in order to avoid gaps or highly overlapping MFs in fuzzy partition.

Here, this is problem is addressed as follows. First, the set of those input variables which use different MFs in the rules than before this operator is determined. After that $M_A$ for each of these input variables is determined. For those input variables for which $M_A \geq 2$, new unevenly distributed partition with $M_A$ MFs is created by the partition algorithm in Section 4.4. If $M_A < 2$, all non-zero conditions, if any, of that input variable are forced to zero. Therefore, it is guaranteed that even when the number of MFs in input variable or the number of input variables in FM is altered, all fuzzy partitions remain transparent.

Operator 4 – Rule Consequent Learning: This operator modifies a consequent $s_i, i = 1, \ldots, R$ (see Section 3.2.1) of a randomly selected active rule by replacing it by random consequent chosen from $[1, M_{\text{out}}]$. A rule is active if it has at least one non-zero rule condition.

Operator 5 – Static Output MFs Tuning: This operator modifies the lateral
displacement of a randomly selected active output MF center $o_i, i = 1, \ldots, M_{out}$ (see Section 3.2.1). An output MF is active if it is used in at least one of the active rules. The selected output MF center is then modified, such that, the static constraints in Section 4.3 are satisfied.

Crossover Operators

**Operator 1 – Dynamic MFs Tuning:** This operator is similar to mutation operator 1 and it also modifies input MF parameters using dynamic MFs tuning. The difference between these operators is that the mutation operator is unary, whereas here two FMs are selected as parents and BLX-0.5 crossover [8, 41] is applied twice resulting to two FMs (i.e. children) in which the MF parameter values depend on the difference between the parameter values of the parents. If the difference is large, this operator mainly explores the search space. On the other hand, if the difference is small, this operator mainly performs fine tuning [7].

**Operator 2 – Swapping of Rule Conditions and MF Parameters:** This operator works on an input variable level. First, it selects randomly an input variable, for which at least one of the parents has at least two active MFs. Then, it pairwisely swaps all rule conditions and input MF parameters of that input variable. So, the child 1 receives all parameters of parent 1, except rule conditions and input MF parameters of the selected input variable, which are received from parent 2. Likewise, child 2 gets all parameters of parent 2, except rule conditions and input MF parameters of the selected input variable, which are received from parent 1. Since both rule conditions and MF parameters of a selected input variable are pairwisely swapped, it is guaranteed that transparency of fuzzy partitions is maintained.

**Operator 3 – Rule Swapping:** This operator is applicable to those rules which are active in at least one of the parents. Out of those rules, $N_{select}$ of them are selected and their rule conditions are pairwisely swapped ($N_{select} \in [1, 5]$ is a random integer).

This operator may alter the number of MFs in input variable and/or the number of input variables in an FM. If that is the case, then the same operators (i.e. par-
tition algorithm or removal of input variable) as with the mutation operator 3 are performed after swapping the rules.

**Operator 4 – Rule Consequent Swapping:** This operator modifies rule consequent of an rule, which is active at least in one of the parents. One of those rules is randomly selected and the rule consequents are pairwisely swapped.

**Operator 5 – Static Output MFs Tuning:** This operator modifies the lateral displacement of output MF centers using BLX-0.5 crossover. It is applicable to those output MF centers which are active in both of the parents. One of them is randomly selected from both parents (the same from both parents) and modified, such that, the static constraints in Section 4.3 are satisfied.

### 4.7 Summary

The purpose of the second MGFS proposals was to improve search ability and to improve transparency of fuzzy partitions compared to the first MGFS proposals. Therefore, the genetic operators of NSGA-II were replaced with specially tailored genetic operators and the rule base was presented with integer-coding. Moreover, new initialization method which guarantees transparency of fuzzy partitions was proposed. It was shown that it overcomes the shortcomings of WM and DT methods. Finally, a dynamic input MFs tuning strategy was introduced. Due to these proposals, search ability is improved and it is guaranteed that the transparency conditions, set by user, are always met by each FM. The following Chapter summarizes and discusses the obtained results for the first and the second MGFS proposals.
Chapter 5

Summary and Discussion of Results

This Chapter summarizes the obtained results using the proposed multiobjective genetic fuzzy systems (MGFSs). Altogether 19 data sets were used and 18 of them can be considered as benchmark problems, since they are publicly available\(^1\). However, some of them present real-world problems. The benchmark data sets are summarized in Table 5.1 and it is seen that six of them are classification problems with different number of input variables, classes, and data points. The rest 12 are regression problems with different number of input variables and data points. The results for classification and regression benchmark problems are summarized in the next two Sections, respectively. After that in Section 5.3 the results of an industrial application, in which the purpose was to identify an accurate and interpretable FM to be used in a bioaerosol detector, are presented. Finally the results are briefly discussed in Section 5.4.

5.1 Results for Benchmark Classification Problems

P1 and P5 dealt with classification problems. The results of P1 are presented next followed by the results of P5 in which the results of these two publications are compared to each other.

\(^1\)The details where they can be obtained are specified in the publications of this thesis.
Table 5.1: Benchmark problems studied in this thesis.

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>Problem Name</th>
<th>Data points</th>
<th>Input variables</th>
<th>Classes</th>
<th>Publication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>Pima Indians diabetes</td>
<td>768</td>
<td>8</td>
<td>2</td>
<td>P1</td>
</tr>
<tr>
<td></td>
<td>Glass</td>
<td>214</td>
<td>9</td>
<td>6</td>
<td>P1, P5</td>
</tr>
<tr>
<td></td>
<td>Wisconsin breast cancer</td>
<td>683</td>
<td>9</td>
<td>2</td>
<td>P1</td>
</tr>
<tr>
<td></td>
<td>Wine</td>
<td>178</td>
<td>13</td>
<td>3</td>
<td>P1, P5</td>
</tr>
<tr>
<td></td>
<td>Cleveland heart disease</td>
<td>297</td>
<td>13</td>
<td>5</td>
<td>P1</td>
</tr>
<tr>
<td></td>
<td>Sonar</td>
<td>208</td>
<td>60</td>
<td>2</td>
<td>P1</td>
</tr>
<tr>
<td>Regression</td>
<td>Electrical Length (Ele1)</td>
<td>495</td>
<td>2</td>
<td>-</td>
<td>P4</td>
</tr>
<tr>
<td></td>
<td>Electrical Maintenance (Ele2)</td>
<td>1056</td>
<td>4</td>
<td>-</td>
<td>P3, P4</td>
</tr>
<tr>
<td></td>
<td>Mackey-Glass (MG)</td>
<td>500</td>
<td>4</td>
<td>-</td>
<td>P3, P4</td>
</tr>
<tr>
<td></td>
<td>Lorenz</td>
<td>500</td>
<td>4</td>
<td>-</td>
<td>P3, P4</td>
</tr>
<tr>
<td></td>
<td>Abalone (ABA)</td>
<td>4177</td>
<td>8</td>
<td>-</td>
<td>P4</td>
</tr>
<tr>
<td></td>
<td>Weather Izmir (IZ)</td>
<td>1461</td>
<td>9</td>
<td>-</td>
<td>This overview</td>
</tr>
<tr>
<td></td>
<td>Weather Ankara (WA)</td>
<td>1609</td>
<td>9</td>
<td>-</td>
<td>This overview</td>
</tr>
<tr>
<td></td>
<td>Box-Jenkins Gas Furnace (Gas)</td>
<td>290</td>
<td>10</td>
<td>-</td>
<td>P3, P4</td>
</tr>
<tr>
<td></td>
<td>Treasury (TR)</td>
<td>1049</td>
<td>15</td>
<td>-</td>
<td>P4</td>
</tr>
<tr>
<td></td>
<td>Mortgage (MO)</td>
<td>1049</td>
<td>15</td>
<td>-</td>
<td>P4</td>
</tr>
<tr>
<td></td>
<td>Computer Activity (CA)</td>
<td>8192</td>
<td>21</td>
<td>-</td>
<td>P4</td>
</tr>
<tr>
<td></td>
<td>Pole Telecommunication (PT)</td>
<td>15000</td>
<td>26</td>
<td>-</td>
<td>This overview</td>
</tr>
</tbody>
</table>

5.1.1 P1: Comparison of Two MGFSs

In P1 the proposed MGFS was validated using six classification problems shown in Table 5.1. 10-fold crossvalidation was repeated 10 times (i.e. $10 \times 10$-CV) for each problem. Therefore, the proposed MGFS was run altogether 100 runs for each problem.

The averaged results were mainly compared to another MGFS [51] which also utilized NSGA-II algorithm and formulated the multiobjective optimization problem (MOP) in three different ways; MOP-1 was used to maximize the accuracy and to minimize the number of rules. MOP-2 was applied to maximize the accuracy and to minimize the number of rule conditions. Finally, MOP-3 was used to maximize the accuracy, minimize the number of rule conditions and to minimize the number of rules. The MOP-3 used the same objective functions as P1. The comparative
MGFS performed rule learning but did not tune the parameters of MFs. Instead, the MFs for rules were selected from four different partitions with two, three, four, and five MFs. Because "don’t care" MF can also be selected, there were altogether 15 rule condition candidates.

In addition, the results were compared to six C4.5 decision tree (DT) variants, representing different splitting strategies and evaluation functions [35]. The comparative approaches [35, 51] used the same experimental setup as in P1 (i.e. $10 \times 10$-CV). Moreover, the comparative MGFS [51] used the same amount of fitness evaluations as the MGFS in P1.

The averaged best test accuracy among the final population of fuzzy models (FMs) obtained by the MGFSs were compared together with the averaged test accuracy of the generated DTs. It was seen that the MGFSs lead to better accuracy than the DTs. Also, the MGFS proposed in this thesis, led into more accurate FMs than the comparative MGFS in five out of six data sets.

The averaged Pareto fronts were also compared. The Pareto fronts depicted the test accuracy of FMs with different number of rules and rule conditions. It was found out that on three of the studied data sets none of the FMs of the proposed MGFS were dominated by the FMs of the comparative study and some of them

**Figure 5.1:** Glass data set: A comparison between the MGFS proposed in P1 and the MGFSs MOP-1, MOP-2, and MOP-3 proposed in [51]. Some of the FMs obtained by the MGFS in P1 dominate some of the FMs obtained by the comparative MGFS. This Figure is adapted from P1.
dominated the FMs of the comparative study. An example of this is shown in Fig. 5.1, in which the averaged Pareto fronts for Glass data set are shown. On the rest three data sets, some FMs of the proposed MGFS were dominated by the FMs of the comparative MGFS and vice versa.

In this comparison the interpretability was measured by number of rules and rule conditions. Transparency of fuzzy partitions was not discussed. Since three parameter MFs tuning with static constraints was used in P1, transparency of fuzzy partitions is not guaranteed. In the comparative study [51] the MFs were taken from four different partitions which by themselves are strong fuzzy partitions and therefore considered highly interpretable. Nevertheless, when taking MFs from different partitions, it is possible that the resulting global partition is not always transparent [72]. However, according to the opinion of the author of this thesis, the fuzzy partitions in the comparative study are more transparent on average than the fuzzy partitions in P1.

The number of distinct Pareto optimal FMs in the final population was also compared. It was noticed that the proposed MGFS usually obtains less distinct Pareto optimal FMs than the MGFS of the comparative study. That was probably due to the initialization method which selected the relevant input variables and therefore reduced the number of distinct FMs. On the other hand, it also eased the further optimization by NSGA-II. Finally, when the number of input variables is reduced, the computational costs related to fitness evaluation are also reduced.

5.1.2 P5: Comparison of Dynamic and Static MF Tuning

The MGFS in P5, which uses the dynamic MFs tuning strategy was compared to the former MGFS in P1, which uses the MFs tuning strategy with static constraints. As an experimental setup 10×10-CV was used and Wine and Glass classification problems were studied. The same amount of fitness evaluations was used in both publications.

The results were rather interesting. It was expected that the former MGFS in P1 should lead into more accurate results, since it does not guarantee transparency of fuzzy partitions. That is because it was shown in P4 that there is a trade-off
between accuracy and transparency of fuzzy partitions. However, for both of the studied problems there was no statistical difference in test accuracy according to Welch $t$-test [80, 91]. That indicates that the proposed techniques, namely initialization method, dynamic tuning strategy, and the genetic operators, work efficiently together obtaining FMs with good accuracy while maintaining the transparency of fuzzy partitions.

The difference between the fuzzy partitions obtained by the MGFSs in P1 and P5 is seen in Figs. 5.2 and 5.3. Clearly, the fuzzy partitions obtained by the MGFS in P5 (Fig. 5.3) are more transparent than the fuzzy partitions obtained by the MGFS in P1 (Fig. 5.2). One can notice from these Figures that both MGFSs not only tune the MFs but also perform granularity learning and input variable selection.
Figure 5.3: Glass problem: An example of fuzzy partitions obtained by the MGFS in P5. All partitions are transparent. Membership degree is denoted by $\mu$. This Figure is modified from P5.

The MGFS in P5 used on average less input variables than the MGFS in P1.

5.2 Results for Benchmark Regression Problems

P3 and P4 dealt with regression problems and the obtained results are summarized in the next two Sections. P3 compared its results to two state-of-the-art MGFSs but this comparison was rather limited. On the other hand, the MGFS in P4 was rigorously evaluated against several recent approaches.
5.2.1 P3: Comparative Study of Three MGFSs

Lorenz, Mackey-Glass (MG), Electrical Maintenance (Ele2), and Box-Jenkins Gas Furnace (Gas) problems were studied in P3. As an experimental setup, 6×5-CV was used. The results were compared to two MGFSs [4, 19]. The obtained results were encouraging in a sense that the proposed MGFS obtained more compact and accurate FMs than the comparative MGFSs. It must be, however, pointed out that in [19] strong fuzzy partitions without MFs tuning were used, which means that the fuzzy partitions in [19] are much more transparent than the fuzzy partitions in P3. Also, in [4] much less fitness evaluations was used than in P3. Therefore, superiority between the MGFSs was not clearly shown.

5.2.2 P4: Rigorous Performance Evaluation

P4 evaluated the performance of the proposed MGFS against eleven recent multiobjective and mono-objective GFSs, including the MGFS proposed in P3. In this overview, a recent MGFS [7] is also included to this comparison. The comparative approaches are shown in Table 5.2 and it is seen that all of them apply MFs tuning, however, with different tuning strategies. Based on the tuning strategy, it was evaluated how transparent the resulting fuzzy partitions on average are. Thus, in this study interpretability of FMs is not simply evaluated by quantitative measures, such as, number of rules and rule conditions, but transparency of fuzzy partitions is also an aspect of interpretability.

As an experimental setup 6×5-CV was used in P4, in this overview, and in all comparative studies. Nine regression data sets, shown in Table 5.1, were studied in P4. However, none of the comparative approaches studied all these data sets. Therefore, for each data set, the results were compared to those approaches for which there were published results available. Additional three data sets, Weather Ankara (WA), Weather Izmir (IZ), and Pole Telecommunication (PT), were included in this overview in order to perform the results comparison on nine data sets against the MGFS in [7]. For the sake of fairness, the number of fitness evaluations was altered in P4, such that, the same amount as in the comparative studies was used. In the new experiments performed in this overview the number of fitness evaluations is set to one third of the fitness evaluations used in the
Table 5.2: Properties of the comparative multiobjective and mono-objective GFSs. GL, GL+S, GLA, GLA+S are mono-objective GFSs, whereas the rest are MGFSs. This Table is adapted from P4.

<table>
<thead>
<tr>
<th>Ref. (Year)</th>
<th>Name</th>
<th>MFs tuning</th>
<th>Rule selection</th>
<th>Rule learning</th>
<th>Input variable selection: Initialization Learning</th>
<th>Transparency of fuzzy partition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GL+S</td>
<td>2-tuple</td>
<td>✓</td>
<td></td>
<td>Best</td>
<td>Good</td>
</tr>
<tr>
<td>GLA</td>
<td></td>
<td>3-tuple</td>
<td>✓</td>
<td></td>
<td>Good</td>
<td></td>
</tr>
<tr>
<td>GLA+S</td>
<td></td>
<td>3-tuple</td>
<td>✓</td>
<td></td>
<td>Good</td>
<td></td>
</tr>
<tr>
<td>[8, 37] (2007-9)</td>
<td>TS-NSGA-II</td>
<td>3-parameter + SSI</td>
<td>✓</td>
<td>✓</td>
<td>Average</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS-NSGA-IIAsc</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS-SPEA2</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS-SPEA2Asc</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS-SPEA2Asc2</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3 (2009)</td>
<td></td>
<td>3-parameter + SLI</td>
<td>✓</td>
<td>✓</td>
<td>Poor</td>
<td></td>
</tr>
</tbody>
</table>

SSI and SLI stand for static constrains with small and large variation intervals, respectively.

comparative approach [7]. This can be seen as unfair towards the proposed MGFS because when the number of fitness evaluations increase, the quality of FMs usually increase as well. However, this experimental setup was chosen in order to demonstrate that the proposed MGFS is very competitive even in this situation.

It was shown in P4 that the proposed MGFS always results into at least comparable accuracy and interpretability with the comparative approaches. Moreover, on some benchmark problems it clearly outperformed some of the comparative approaches meaning that the FMs obtained by the proposed approach were more accurate and compact and their fuzzy partitions were more transparent than in the comparative studies. This will be demonstrated in the next two Sections. After that, the MGFS in P4 is compared to the MGFS in P3. Then, the trade-off between accuracy and transparency of fuzzy partitions is discussed. Finally, the results comparison against the MGFS in [7] on nine data sets is performed.
Electrical Maintenance Problem (Ele2)

For this problem with four input variables, the results after 50000 fitness evaluations were compared to [4] and [8], and the results after 100000 fitness evaluations were compared to [37]. That was done in order to use the same amount of fitness evaluations as in the comparative studies. Table 5.3 shows the results. In that Table Initial stands for the results of initial FM and Final is the most accurate FM of the population after the MGFS optimization. R, R_{cond}, MFs, and F stand for the number of rules, rule conditions, input MFs and input variables, respectively. Training and testing mean squared error divided by two is denoted by MSE_{trn} and MSE_{tst}, respectively. Finally, t indicates whether there is significant difference in accuracy according to t-test.\footnote{With 95\% confidence. + indicates the best averaged result in the column, + means that the performance of the corresponding row is worse than the best result, and = means that there is no significant statistical difference compared to the best result.}

It is interesting to compare the FMs obtained by the proposed initialization method to the FMs initialized by WM, since all the comparative MGFSs in [4, 8, 37] were initialized by it. It can be seen from Table 5.3 that WM algorithm leads to more accurate FMs in this case. However, the FMs obtained by WM algorithm are significantly more complex. Therefore, the number of parameters to be optimized is high and the computational costs are increased because fitness evaluations of complex FMs are computationally taxing. An interesting aspect is also that the FM initialized by the proposed method is more compact than the FMs of the comparative studies [4, 8, 37] after the MGFSs optimization.

When the results after 50000 fitness evaluations are compared to [4] and [8], it can be seen that Final is the most accurate in both training and testing sets. Furthermore, there is statistical difference between it and all the comparative MGFSs when testing accuracy is considered. Moreover, Final is clearly more compact than the comparative FMs and its fuzzy partitions are at least at the same level as in the comparative studies [4] and [8], which can be seen from Table 5.2.

The results after 100000 fitness evaluations show that Final is significantly more accurate in both training and test sets than the most accurate FM of the comparative approach [37]. Also, it is clearly more compact having less rules, rule conditions and input variables. Finally, when transparency of fuzzy partitions is
Table 5.3: Results comparison for Ele2 problem. The best results per column in each category (i.e. initialization, after 50000 and 100000 evaluations) are shown in boldface. t-test is also applied to each category separately. This Table is modified from P4.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Method</th>
<th>R</th>
<th>R_cond</th>
<th>MFs</th>
<th>F</th>
<th>MSE_trn</th>
<th>σ_trn</th>
<th>t</th>
<th>MSE_tst</th>
<th>σ_tst</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initialization methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4, 8, 37]</td>
<td>WM</td>
<td>65.0</td>
<td>260.0</td>
<td>N/A</td>
<td>4.0</td>
<td>57605</td>
<td>2841</td>
<td>*</td>
<td>57934</td>
<td>4733</td>
<td>*</td>
</tr>
<tr>
<td>P4</td>
<td>Initial</td>
<td><strong>26.2</strong></td>
<td><strong>61.6</strong></td>
<td>15.4</td>
<td><strong>3.4</strong></td>
<td>74719</td>
<td>15065</td>
<td>+</td>
<td>74274</td>
<td>12069</td>
<td>+</td>
</tr>
<tr>
<td>After 50000 fitness evaluations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4]</td>
<td>GL</td>
<td>65.0</td>
<td>260.0</td>
<td>N/A</td>
<td>4.0</td>
<td>23064</td>
<td>1479</td>
<td>+</td>
<td>25654</td>
<td>2611</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>GLA</td>
<td>65.0</td>
<td>260.0</td>
<td>N/A</td>
<td>4.0</td>
<td>17950</td>
<td>1889</td>
<td>+</td>
<td>21212</td>
<td>2686</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>GLA+S</td>
<td>49.4</td>
<td>197.6</td>
<td>N/A</td>
<td>4.0</td>
<td>17538</td>
<td>2391</td>
<td>+</td>
<td>21491</td>
<td>4168</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>GL+S</td>
<td>49.1</td>
<td>196.4</td>
<td>N/A</td>
<td>4.0</td>
<td>18801</td>
<td>2669</td>
<td>+</td>
<td>22586</td>
<td>3550</td>
<td>+</td>
</tr>
<tr>
<td>[8]</td>
<td>TS-NSGA-II</td>
<td>48.1</td>
<td>192.4</td>
<td>N/A</td>
<td>4.0</td>
<td>16321</td>
<td>1636</td>
<td>+</td>
<td>20423</td>
<td>3138</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>TS-NSGA-II</td>
<td>41.0</td>
<td>164.0</td>
<td>N/A</td>
<td>4.0</td>
<td>14488</td>
<td><strong>965</strong></td>
<td>+</td>
<td>18419</td>
<td>3054</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>TS-SPEA2Acc</td>
<td>34.5</td>
<td>138.0</td>
<td>N/A</td>
<td>4.0</td>
<td>11081</td>
<td>1186</td>
<td>=</td>
<td>14161</td>
<td>2191</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>TS-SPEA2</td>
<td>33.0</td>
<td>132.0</td>
<td>N/A</td>
<td>4.0</td>
<td>13272</td>
<td>1265</td>
<td>+</td>
<td>17533</td>
<td>3226</td>
<td>+</td>
</tr>
<tr>
<td>P4</td>
<td>Final</td>
<td><strong>25.0</strong></td>
<td><strong>51.5</strong></td>
<td>14.3</td>
<td><strong>3.1</strong></td>
<td>10861</td>
<td>1436</td>
<td>*</td>
<td>12336</td>
<td><strong>2065</strong></td>
<td>*</td>
</tr>
<tr>
<td>After 100000 fitness evaluations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[37]</td>
<td>TS-SPEA2Acc²</td>
<td>29.8</td>
<td>119.2</td>
<td>N/A</td>
<td>4.0</td>
<td>10325</td>
<td>1121</td>
<td>+</td>
<td>13935</td>
<td>2759</td>
<td>+</td>
</tr>
<tr>
<td>P4</td>
<td>Final</td>
<td><strong>24.9</strong></td>
<td><strong>49.1</strong></td>
<td>14.3</td>
<td><strong>3.1</strong></td>
<td>9366</td>
<td>887</td>
<td>*</td>
<td>10429</td>
<td><strong>1646</strong></td>
<td>*</td>
</tr>
</tbody>
</table>

considered, it can be seen from Table 5.2 that the fuzzy partitions in P4 are more transparent than in the comparative study.

**Abalone Problem**

According to [37], this problem has a high noise level and different MGFSs usually lead to somewhat similar accuracy. However, complexity of the FMs obtained by different MGFSs may vary. The results for the initial FMs obtained by WM method and by the proposed initialization method are shown in Table 5.4. In this case, the proposed initialization method obtains slightly more accurate initial FMs than the WM algorithm. However, t-test shows no statistical significance between the methods. The FMs obtained by the proposed initialization method
Table 5.4: Results comparison for Abalone problem. This Table is modified from P4.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Method</th>
<th>R</th>
<th>$R_{\text{cond}}$</th>
<th>MFs</th>
<th>$F$</th>
<th>MSE$_{\text{trn}}$</th>
<th>$\sigma_{\text{trn}}$</th>
<th>t</th>
<th>MSE$_{\text{tst}}$</th>
<th>$\sigma_{\text{tst}}$</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>[37]</td>
<td>WM</td>
<td>68.2</td>
<td>545.6</td>
<td>N/A</td>
<td>8.0</td>
<td>8.407</td>
<td>0.443</td>
<td>=</td>
<td>8.422</td>
<td>0.545</td>
<td>=</td>
</tr>
<tr>
<td>P4</td>
<td>Initial</td>
<td>30.0</td>
<td>130.6</td>
<td>16.4</td>
<td>6.6</td>
<td>7.946</td>
<td>2.879</td>
<td>*</td>
<td>7.906</td>
<td>2.742</td>
<td>*</td>
</tr>
</tbody>
</table>

After 100000 fitness evaluations

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Method</th>
<th>R</th>
<th>$R_{\text{cond}}$</th>
<th>MFs</th>
<th>$F$</th>
<th>MSE$_{\text{trn}}$</th>
<th>$\sigma_{\text{trn}}$</th>
<th>t</th>
<th>MSE$_{\text{tst}}$</th>
<th>$\sigma_{\text{tst}}$</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>[37]</td>
<td>TS-SPEA2$_{\text{Acc}}$</td>
<td>22.2</td>
<td>177.6</td>
<td>N/A</td>
<td>8.0</td>
<td>2.368</td>
<td>0.085</td>
<td>*</td>
<td>2.511</td>
<td>0.263</td>
<td>=</td>
</tr>
<tr>
<td>P4</td>
<td>Final</td>
<td>20.5</td>
<td>43.3</td>
<td>10.7</td>
<td>4.4</td>
<td>2.389</td>
<td>0.074</td>
<td>=</td>
<td>2.423</td>
<td>0.173</td>
<td>*</td>
</tr>
</tbody>
</table>

are, nonetheless, clearly more compact than the initial FMs obtained by WM.

When the FMs after MGFS optimization are compared, it is seen that there is no significant difference in accuracy. The difference in complexity is, however, clear. The FMs obtained by the proposed MGFS have much less rule conditions and input variables. Moreover, according to Table 5.2, the fuzzy partitions of the FMs obtained by the proposed MGFS can be considered more transparent.

Comparison Between P4 and P3

The MGFS in P4 was also compared to MGFS in P3 on MG, Lorenz and Gas problems with the same amount (210000) of fitness evaluations. It was seen that MGFS in P4 leads to significantly more accurate FMs which, however, contain much more rules and rule conditions than the FMs in P3. Finally, the FMs in P4 generally have much better transparency of fuzzy partitions than the FMs in P3.

Trade-off between Transparency of Fuzzy Partitions and Accuracy

The effect of parameters $\alpha$, $\beta$, and $\gamma$ which directly determines the transparency of fuzzy partitions was studied in P4 as well. Experiments with $\alpha = 0.8$, $\beta =$
0.05, and γ = 0.25 and α = 0.6, β = 0.4, and γ = 0.1 using the 6×5-CV were performed. It was clearly shown that by increasing the transparency of fuzzy partitions the accuracy is deteriorated (see also Fig. 5.4).

The importance of including the transparency of fuzzy partitions as an aspect of interpretability is illustrated in Figs. 5.5 and 5.6. In those Figures FMs of a single run of Mortgage problem with different settings of α, β and γ are shown. Those FMs are the most accurate FMs of the final Pareto fronts. The one in Fig. 5.5 uses three input variables and its rule base consists of 11 rules and 23 rule conditions. Its training and testing accuracies are \( \text{MSE}_{\text{trn}} = 0.028, \text{MSE}_{\text{tst}} = 0.072 \). The other one in Fig. 5.6 uses four input variables and its rule base consists of 11 rules and 25 rule conditions. Its training and testing accuracies are \( \text{MSE}_{\text{trn}} = 0.036, \text{MSE}_{\text{tst}} = 0.090 \). The first one clearly dominates the second one, because it is both more accurate and more compact. It should be preferred if the decision is based on quantitative criteria only. However, if the qualitative criteria about the transparency of fuzzy partitions is included, the decision about the superiority is not so clear. That is because the second one has more transparent fuzzy partitions and might be preferred by an user for which this aspect is of importance.
Figure 5.5: Mortgage problem: An example FM with $\alpha = 0.8$, $\beta = 0.05$, $\gamma = 0.25$. Membership degree is denoted by $\mu$. This Figure is adapted from P4.

Figure 5.6: Mortgage problem: An example FM with $\alpha = 0.6$, $\beta = 0.4$, $\gamma = 0.1$. Membership degree is denoted by $\mu$. This Figure is adapted from P4.
Results Comparison on Nine Data Sets

At the time of writing the first versions of P4, there existed no published results for a MGFS on several data sets. Therefore, it was necessary to compare the proposed MGFS against different MGFS on different problems. At the time of writing the final version of P4, the results of a MGFS [7], denoted by PAES\textsubscript{KB}, on nine data sets became available; however, at that time there was no time to perform new experiments. Now those experiments are performed. Results comparison on multiple data sets enables application of non-parametric statistical tests to measure the difference between the MGFSs. As a non-parametric test, Wilcoxon signed rank test with 95% confidence is used here. This test is more reliable than \( t \)-test in a sense that it does not assume normal distribution. Moreover, exceptionally bad or good performance on few problems (i.e. outliers) does not affect this test as much as it affects \( t \)-test. Detailed information about this test is available in [31,92].

Like mentioned previously, the number of fitness evaluations is selected here as one third of the fitness evaluations used in the comparative study [7]. So, the proposed MGFS was run with 100000 fitness evaluations whereas the comparative approach was run 300000 fitness evaluations. The number of input and output MFs were selected the same as in the comparative study. The parameters \( \alpha, \beta, \gamma \) were set as 0.7, 0.3, 0.2, respectively. With these settings, the membership value at any intersection of two MFs is between 0.3 and 0.7. In the middle of an MF, another MF can receive the membership value at most 0.2. For the comparative approach, the membership value at any intersection of two MFs is between 0 and 1, whereas in the middle of an MF, the maximum membership value another MF can receive is almost 1. So in this sense, our approach maintains better transparency of fuzzy partitions. However, the shape of the MFs is fixed in the comparative approach which is sometimes considered as a positive aspect with regards to transparency of fuzzy partitions. But in any case, the applied settings should be considered fair towards the comparative approach.

Table 5.5 shows the averaged results for the most accurate FMs of the Pareto front. It can be seen that the proposed MGFS obtains better test accuracy in seven out of nine data sets. It also obtains FMs which use less input variables in all data sets except in Ele1. That data set, however, has only two input variables which makes it difficult to reduce dimensionality without deteriorating the accuracy too.
Table 5.5: Results comparison between the MGFSs in P4 and in [7]. The MGFSs in P4 and in [7] were run 100000 and 300000 fitness evaluations, respectively.

<table>
<thead>
<tr>
<th></th>
<th>MGFS proposed in P4</th>
<th></th>
<th>PAES\textsubscript{KB} in [7]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( R )</td>
<td>( R_{\text{cond}} )</td>
<td>( F )</td>
</tr>
<tr>
<td>Ele1</td>
<td>13</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>Ele2</td>
<td>24</td>
<td>45</td>
<td>3</td>
</tr>
<tr>
<td>ABA</td>
<td>21</td>
<td>41</td>
<td>4</td>
</tr>
<tr>
<td>IZ</td>
<td>28</td>
<td>61</td>
<td>5</td>
</tr>
<tr>
<td>WA</td>
<td>27</td>
<td>55</td>
<td>4</td>
</tr>
<tr>
<td>TR</td>
<td>28</td>
<td>58</td>
<td>4</td>
</tr>
<tr>
<td>MO</td>
<td>28</td>
<td>66</td>
<td>5</td>
</tr>
<tr>
<td>CA</td>
<td>27</td>
<td>65</td>
<td>7</td>
</tr>
<tr>
<td>PT</td>
<td>24</td>
<td>74</td>
<td>7</td>
</tr>
</tbody>
</table>

much. For low-dimensional problems, the number of rules and rule conditions in FMs obtained by the proposed MGFS is less than in FMs obtained by PAES\textsubscript{KB}. However, for high-dimensional problems the FMs obtained by PAES\textsubscript{KB} have less rules and rule conditions. In order to apply Wilcoxon signed rank test and to see whether there is significant difference in test accuracy, normalized difference between the test accuracies is computed according to [7]:

\[
\text{DIFF} = \frac{\text{MSE}_{\text{tst}}(\text{PAES}_{\text{KB}}) - \text{MSE}_{\text{tst}}(\text{P4})}{\text{MSE}_{\text{tst}}(\text{PAES}_{\text{KB}})}. \tag{5.1}
\]

The normalized differences are shown in Table 5.6. Those values are then ranked according to their absolute magnitude and the result is shown in the same Table. After that, the sum of ranks for which the MGFS in P4 outperformed PAES\textsubscript{KB} is computed and denoted by \( R^+ \). Likewise, the sum of ranks for which PAES\textsubscript{KB} outperformed the MGFS in P4 is computed and denoted by \( R^- \). Those values are shown in Table 5.7. Now, Wilcoxon signed rank test can be applied and it is seen that the null hypothesis is rejected (p-value < 0.05) meaning that there is significant difference in test accuracy between the approaches. Hence, it was
Table 5.6: Comparison of running times and test accuracies. DIFF is the normalized difference in test accuracy. The MGFSs in P4 and in [7] were run 100000 and 300000 fitness evaluations, respectively.

<table>
<thead>
<tr>
<th>Problem (F / Data points)</th>
<th>DIFF</th>
<th>Rank</th>
<th>Time (min)</th>
<th>Time(PAES\textsubscript{KB}) / Time(P4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ele1 (2/495)</td>
<td>-0.0021</td>
<td>-1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Ele2 (4/1056)</td>
<td>0.0343</td>
<td>3</td>
<td>7</td>
<td>27</td>
</tr>
<tr>
<td>ABA (8/4177)</td>
<td>0.0161</td>
<td>2</td>
<td>24</td>
<td>131</td>
</tr>
<tr>
<td>IZ (9/1461)</td>
<td>0.2483</td>
<td>5</td>
<td>14</td>
<td>55</td>
</tr>
<tr>
<td>WA (9/1609)</td>
<td>0.4643</td>
<td>6</td>
<td>15</td>
<td>69</td>
</tr>
<tr>
<td>TR (15/1049)</td>
<td>0.5714</td>
<td>8</td>
<td>9</td>
<td>61</td>
</tr>
<tr>
<td>MO (15/1049)</td>
<td>0.7778</td>
<td>9</td>
<td>10</td>
<td>64</td>
</tr>
<tr>
<td>CA (21/8192)</td>
<td>0.4944</td>
<td>7</td>
<td>86</td>
<td>634</td>
</tr>
<tr>
<td>PT (26/15000)</td>
<td>-0.1011</td>
<td>-4</td>
<td>121</td>
<td>921</td>
</tr>
</tbody>
</table>

Table 5.7: Application of Wilcoxon signed rank test with 95% confidence. Null hypothesis is rejected in favor of the MGFS in P4.

<table>
<thead>
<tr>
<th>Methods</th>
<th>R\textsuperscript{+}</th>
<th>R\textsuperscript{-}</th>
<th>Null hypothesis</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAES\textsubscript{KB} vs. P4</td>
<td>40</td>
<td>5</td>
<td>Rejected</td>
<td>0.0391</td>
</tr>
</tbody>
</table>

shown that the MGFS in P4 outperforms PAES\textsubscript{KB} in test accuracy even though it uses only one third of the fitness evaluations used by PAES\textsubscript{KB}.

The averaged running times for each problem are reported in Table 5.6. They show how long it takes to identify a set of Pareto-optimal FMs from data (i.e. times include both the initialization and the further optimization parts). It is seen that the proposed MGFS uses less time and the difference between the running times is clear especially for problems with many input variables and data points. It must be, however, kept in mind that different amount of fitness evaluations and different computers were used in the experiments. The proposed MGFS was run on a laptop which has Intel Core 2 Duo CPU at 2.40 GHz with 2 GB of RAM and only one of
Table 5.8: Results comparison between the MGFSs in P4 and in [7]. Both MGFSs were run 300000 fitness evaluations.

<table>
<thead>
<tr>
<th>Data</th>
<th>MGFS proposed in P4</th>
<th>PAESKB in [7]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R$ $R_{cond}$ $F$ MFs $\text{MSE}<em>{\text{trn}}$ $\sigma</em>{\text{trn}}$ $\text{MSE}<em>{\text{tst}}$ $\sigma</em>{\text{tst}}$ Time (min)</td>
<td>$R$ $R_{cond}$ $F$ $\text{MSE}_{\text{tst}}$ Time (min)</td>
</tr>
<tr>
<td>Ele1</td>
<td>13 21 2 9 151994 8438 193447 33203 9</td>
<td>27 46 2 194028 4</td>
</tr>
<tr>
<td>Ele2</td>
<td>22 41 3 14 8956 771 9925 1169 22</td>
<td>30 65 4 12606 27</td>
</tr>
<tr>
<td>ABA</td>
<td>19 42 4 11 2.35 0.09 2.41 0.20 57</td>
<td>29 107 8 2.48 131</td>
</tr>
<tr>
<td>IZ</td>
<td>24 48 5 18 0.91 0.31 1.01 0.34 36</td>
<td>25 91 9 1.49 55</td>
</tr>
<tr>
<td>WA</td>
<td>25 46 4 16 1.61 0.68 1.89 1.04 37</td>
<td>28 103 9 3.92 69</td>
</tr>
<tr>
<td>TR</td>
<td>25 52 4 16 0.03 0.00 0.05 0.03 24</td>
<td>11 40 12 0.14 61</td>
</tr>
<tr>
<td>MO</td>
<td>26 58 5 17 0.01 0.00 0.02 0.01 27</td>
<td>12 49 14 0.09 64</td>
</tr>
<tr>
<td>CA</td>
<td>22 52 7 21 5.00 0.49 5.21 0.65 193</td>
<td>10 30 12 13.43 634</td>
</tr>
<tr>
<td>PT</td>
<td>19 61 7 17 88 45 88 47 284</td>
<td>14 53 19 89 921</td>
</tr>
</tbody>
</table>

the cores was used in the experiments. The comparative MGFS was run on Intel Core 2 Quad CPU at 2.50 GHz with 8 GB of RAM and only one of the cores was used. Another aspect which affects the computational costs is the programming language. The proposed MGFS was implemented partly in MATLAB and partly in C. C4.5 and NSGA-II consist of C-code written by Quinlan [78] and Deb et al. [30], respectively. The fitness evaluation part, which is usually the most time consuming part of the MGFS procedure, was written in C by the author of this thesis. Genetic operators and part of the initialization procedure, such as WM method, were written in MATLAB (as m-file) by the author of this thesis. In [7] the programming language was not specified.

Since the comparative approach obtained better test accuracy in Ele1 and PT problems, it is studied whether the proposed MGFS would lead to better accuracy if the number of fitness evaluations was increased from 100000 to 300000, like in the comparative study. Table 5.8 shows the results for all nine problems after 300000 fitness evaluations and it is seen that with these settings the proposed MGFS obtains better test accuracy in all problems. One can also notice by comparing the results in Tables 5.5 and 5.8 that, in general, the accuracy and compactness of the
FMIs obtained by the proposed MGFS are improved when the number of fitness evaluations is increased. The running times are naturally increased due to increment in the number of fitness evaluations as well. However, the proposed MGFS is still faster than the comparative MGFS in all problems except Ele1, which is the problem with least input variables and data points. For that problem, the proposed MGFS is not able to gain speedup by reducing the number of input variables because there are only two input variables.

5.3 Results of an Industrial Application

In P2 the aim was to create an automatic reasoning mechanism (i.e. an FM) for a bioaerosol detector developed in [57] and shown in Fig. 5.7. The main requirements for the reasoning mechanism were low false positive (FP) rate, high true positive (TP) rate and interpretability. The schemata of the reasoning procedure is shown in Fig. 5.8. As seen, the purpose is to deduce whether there are harmful or safe bioaerosols in the air based on the background and UV-fluorescence measurements. Altogether the reasoning is based on four input variables. The number of data points is 10268 and there are two classes, namely "harmful" and "safe". Due to confidentiality of the project, details about the bioaerosol detector and its signals need to be omitted.

The data were collected as a result of two measurement campaigns during autumn 2006. First of them was performed in the Umeå trial field in Sweden and the second in the laboratory of Finnish Defence Forces Technical Research Center in Finland. The distribution of the collected data was skewed; about 80% of data points presented safe bioaerosols and the rest 20% harmful bioaerosols. That was the main reason why accuracy was measured by FP and TP rates.

When dealing with industrial data, data preprocessing is naturally essential. Thus, before applying the proposed MGFS, the data were filtered, features were extracted, and some actions to correct faulty data were needed. Despite the challenging data, the proposed MGFS was able to find FMIs presenting good trade-offs between the objectives. For example, an FM with $TP = 0.908$, $FP = 0.051$ on test data was obtained. That accurate FM had also transparent fuzzy partitions (transparency index $T = 0.003$). Moreover, it was very compact, having only seven...
During the project, the bioaerosol detector was also further improved. Some of the guidelines for further improvement were given through an analysis of the FM on the collected data. For example, if a false alarm was given, the rule which caused it was determined. Then, by tracing back to the rule conditions and MFs of that rule, it was seen what are the measurement values at that given time. Sometimes, those measurement values were not what they were supposed to be and therefore a dysfunction in the bioaerosol detector was noticed. Other times, the false alarm was due to the FM itself. Therefore, depending on the source of the false alarm, either the bioaerosol detector or the FM was further developed.

As a result of the project, an FM was installed in January 2007 into the bioaerosol detector and the preliminary tests were positive. After that, the bioaerosol detector was still further developed, which changed the absolute values of the measurements. Therefore, update of the FM was necessary as well. At this point, however, there was neither enough data collected by the developed bioaerosol detector nor time to deeply analyze how the absolute values were changed. The measuring mechanism and logic had, however, remained approximately the same.
This Chapter demonstrated the applicability of the proposed MGFSs to six classification and 12 regression benchmark problems. Moreover, the MGFS of P2 was successfully used in an industrial application in which the purpose was to identify an FM to be used in a bioaerosol detector to automatically classify harmful and safe bioaerosols.

Overall, when compared to the state-of-the-art MGFSs in the literature, it was noticed that the proposed MGFSs are very competitive and in some cases they clearly outperform some of the MGFSs in the literature. Wilcoxon signed rank test
also indicated that the MGFS in P4 outperforms a recently proposed MGFS [7] in terms of test accuracy even when it uses only one third of the fitness evaluations used in [7]. Partially due to this, the MGFS in P4 is also faster than the MGFS in [7].

There is, however, still room for improvement. The results indicate that the FMs obtained by the MGFS proposed in P4 have more rules and rule conditions in high-dimensional problems than the FMs in [7]. Moreover, on PT problem with 26 input variables, the accuracy of the FMs obtained by the proposed MGFS is worse than the accuracy of the FMs in [7]. In those experiments, however, the proposed MGFS used only one third of the fitness evaluations used in [7]. By using the same number of fitness evaluations, the proposed MGFS obtained FMs which have better test accuracy in all of the studied problems. Increment in fitness evaluations naturally increased the computational time. Hence, a better solution would be to improve the search ability of the proposed MGFS by modifying its genetic operators. Future work towards this direction is currently under way.

When the second MGFS proposals (P4 and P5) are compared to the first MGFS proposals (P1 and P3), it can be said that the second MGFS proposals are superior\(^3\). That is because they generally lead to good accuracy and compactness and always guarantee that the fuzzy partitions meet the transparency criteria set by user. On the other hand, transparency of fuzzy partitions was not guaranteed in P1 and P3. The MGFS of P2 was not compared to the second MGFS proposals in any of the publications; however, the second MGFS proposals were developed because the performance of P2 was not satisfactory when experimented on some data sets. During the development, it was noticed that the second MGFS proposals clearly improve the results.

Different MGFSs were compared in this thesis as a whole. They consist of several individual components, such as, initialization method and MFs tuning strategy. The performance of their individual components was usually not evaluated against each other. Only the initialization methods were compared in Section 5.2.2 but not rigorously. This is to say that based on the results comparisons, it cannot be reliably stated that, for example, the proposed dynamic MFs tuning strategy outperforms another tuning strategy because by applying different initialization

\(^3\)Of course this is somewhat subjective because sometimes the first MGFS proposals may obtain, for example, more compact FMs than the second MGFS proposals.
methods, performance of a particular MGFS can be deteriorated or improved. This is because appropriate initialization reduces the search space and therefore eases the derivation of better FMs [3]. So, from the obtained results only the superiority between certain MGFSs as a whole can be reliably established. That, however, is often the goal since users usually desire to find the best performing MGFS for a given problem. Nonetheless, rigorous evaluation of individual components is important since it may reveal some of their weaknesses. When they are improved or replaced, it may lead to improvement in MGFS as a whole. This topic is left for future study.
Chapter 6

Conclusions and Further Research Directions

The purpose of this thesis was to develop methods to obtain interpretable and accurate models based on data for classification and regression problems. Fuzzy models (FMs) were selected as a model type because they can be interpretable due to intuitive "If-then" rules. When FMs are derived automatically based on data, interpretability is, however, not guaranteed.

This thesis proposed five multiobjective genetic fuzzy systems (MGFSs) for solving this problem. The conflicting objectives, namely accuracy and interpretability, were subject to maximization using multiobjective optimization with nondominated sorting genetic algorithm II (NSGA-II). The use of MGFSs is beneficial because it enables obtaining not only one trade-off FM, but a Pareto optimal set of FMs presenting different trade-offs between accuracy and interpretability. After the MGFS run, the user selects an FM based on his/her preferences after he/she has seen the available choices.

Identification of interpretable and accurate FMs is a challenging problem especially when the number of input variables is high. The search space is large, which makes it difficult to find good FMs. This thesis tackled this problem mainly by using initialization methods which try to remove irrelevant input variables and therefore to ease the further optimization. Two novel initialization methods were
proposed for that purpose. It was shown that they significantly reduce dimensionality and the number of parameters to be optimized when compared to commonly used Wang-Mendel (WM) initialization method. Moreover, the second initialization proposal, which combines the benefits of WM and decision tree (DT) initialization methods, leads to initial FMs which are of moderate complexity and have transparent fuzzy partitions.

Another challenge is related to membership function (MF) tuning which usually improves accuracy but can deteriorate transparency of fuzzy partitions if appropriate tuning strategy is not applied. Two solutions for that problem were given. First, a novel interpretability index was used to measure the transparency of fuzzy partitions and it was subject to optimization. Second, a dynamic MFs tuning strategy was proposed. It allows the user to specify the level of transparency he/she desires and it guarantees that each FM at any time of the identification procedure meets those requirements. This makes it an unique MFs tuning strategy. Since its underlying idea is rather simple, it can be used to improve other tuning schemes, such as 2-tuple or 3-tuple tuning schemes, as well.

Three of the proposed MGFSs were designed for classification problems and two of them for regression problems. They were evaluated on seven classification and 12 regression problems and compared to several recent GFSs and MGFSs in the literature. The proposed MGFSs perform well in these comparisons and especially the MGFS in P4 shows very good performance against 12 recent approaches. For example, it outperforms a recently proposed MGFS [7] in test accuracy even when it uses only one third of the fitness evaluations used in [7]. Partially because of this, it is also faster than the MGFS in [7]. Finally, it was also shown through an industrial application that MGFSs are suitable for identification of FMs to be used as a reasoning mechanism in a bioaerosol detector.

Although the proposed MGFSs perform well in general, it was pointed out that future research is needed regarding high-dimensional problems and problems with many data points. There are mainly two challenges related to these problems. First, the computational time can be long. Second, it may be hard to find good trade-off FMs when the search space is very large. One idea to reduce the computational costs is to divide the training set and population into subsets and to use parallel distributed implementation with multiple processors [73]. Another idea is to reduce the size of the training set by selecting the most important training
samples [71]. Finally, approximating the fitness function instead of computing its exact value is also a way to reduce the computational costs [20, 59]. Approaches to improve search ability in high-dimensional problems usually reduce the search space by presenting data base with as few parameters as possible and by using heuristic algorithms to aid the search of appropriate rules [16, 38]. Future work towards that direction is currently under way by the author of this thesis.

Besides that, there are also other possibilities to continue the work. For example, deep analysis and rigorous comparison of the individual components of the MGFSs, such as initialization method, coding scheme, MF type, and genetic operators, could be performed. Based on that, possible weaknesses in them could be identified which might lead to improvements in search ability and/or search efficiency. At the moment, the research in developing novel initialization methods is not very active. However, this is probably an important research direction since it has a clear effect on the size of the search space and therefore on the performance of the MGFSs. Moreover, more experiments with the MGFS proposed in P5 could be performed, since it was evaluated only on two data sets. Finally, new industrial application areas could be identified.
Bibliography


© 2007 Elsevier Inc. Reprinted with permission.
Fuzzy classifier identification using decision tree and multiobjective evolutionary algorithms

Pietari Pulkkinen *, Hannu Koivisto

Institute of Automation and Control, Tampere University of Technology, P.O. Box 692, FIN-33101 Tampere, Finland

Received 29 November 2006; received in revised form 3 October 2007; accepted 8 October 2007
Available online 17 October 2007

Abstract

This paper presents a hybrid method for identification of Pareto-optimal fuzzy classifiers (FCs). In contrast to many existing methods, the initial population for multiobjective evolutionary algorithms (MOEAs) is neither created randomly nor a priori knowledge is required. Instead, it is created by the proposed two-step initialization method. First, a decision tree (DT) created by C4.5 algorithm is transformed into an FC. Therefore, relevant variables are selected and initial partition of input space is performed. Then, the rest of the population is created by randomly replacing some parameters of the initial FC, such that, the initial population is widely spread. That improves the convergence of MOEAs into the correct Pareto front. The initial population is optimized by NSGA-II algorithm and a set of Pareto-optimal FCs representing the trade-off between accuracy and interpretability is obtained. The method does not require any a priori knowledge of the number of fuzzy sets, distribution of fuzzy sets or the number of relevant variables. They are all determined by it. Performance of the obtained FCs is validated by six benchmark data sets from the literature. The obtained results are compared to a recently published paper [H. Ishibuchi, Y. Nojima, Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning, International Journal of Approximate Reasoning 44 (1) (2007) 4–31] and the benefits of our method are clearly shown.

Keywords: Fuzzy classifiers (FCs); Multiobjective evolutionary algorithms (MOEAs); Decision trees (DTs); Initialization

1. Introduction

Fuzzy classifiers (FCs) with if-then rules are related to the way human beings think and that is their main advantage over black-box models, such as neural networks. Identification of FCs involves determining the adequate structure and parameters. The structure identification consists of several tasks, such as, selecting the adequate variables, assigning the adequate number of fuzzy sets to each variable and defining the number of fuzzy rules used. In addition to that, the parameters of fuzzy sets need to be specified as well. It was illus-
trated in [1], that such a task is highly complex due to its enormous search space, especially when high-dimen-
sional problems are covered.

Grid-type partitioning is a way to reduce the complexity of the identification problem. In that approach, the
number of fuzzy sets assigned to each variable is fixed to some number and also the parameters of fuzzy sets
are predefined. However that approach suffers from the curse of dimensionality, that is, the number of fuzzy
rules is exponentially increased when the dimensionality of the problem is increased. To overcome that prob-
lem [2] applied grid-type partitioning with “don’t care” linguistic values and selected only the relevant rules
out of the all possible rules. The benefit of the approach is its simple implementation, because it does not modify
the parameters of fuzzy sets. Nevertheless, it was stated in [3] that fuzzy sets are the major components of
FCs, since they affect the accuracy of the model, interpretability of fuzzy rules and also the performance of the
system. It was stated in [2], that homogenously assigned fuzzy sets are intuitive, therefore making the FCs
more interpretable. However, often they do not represent the real distribution of the data and therefore the
accuracy of the obtained FCs is degraded [4,3]. Moreover, the intuitiveness of the linguistic values is also dete-
riorated. To tackle that problem, the fuzzy sets can also be pre-specified by domain experts. However, when
dealing with high-dimensional problems, domain experts will have problems in assigning the fuzzy sets for
each variable. Therefore automatic tuning of the fuzzy sets is usually required.

Recently the goal in FC identification has been in obtaining accurate and interpretable FCs. Naturally
accuracy and interpretability are conflicting objectives. For example, an FC with a vast rule-base may be accu-
rate for training patterns, however, it lacks for interpretability and may not perform well on unseen samples
due to the overfitting. Usually a trade-off between the accuracy and interpretability is sought using evolution-
ary algorithms (EAs) and often those approaches are called genetic fuzzy systems (GFS) [5]. A single trade-off
solution can be found by aggregating multiple objectives (e.g. accuracy, number of rules and number of con-
ditions) into a single fitness function and by setting the weights for each objective [6,7]. However, that requires
work in choosing the appropriate weights, which may be different for each problem at hand. Moreover, it is
not guaranteed that with every run a new solution is found [8]. Since multiobjective evolutionary algorithms
(MOEAs) can find several widely spread Pareto-optimal solutions in a single run without assigning weight val-
ues for each objective, they are often preferred. After a set of solutions is obtained, advances and drawbacks of
them can be considered and a solution can be selected based on the preferences.

When EAs are applied, the population needs to be initialized first. That can be done randomly or manually
like in [9–11]. Adequate initialization, however, can improve the convergence of EAs [12,13]. Hence, it is ben-
eficial to use, for example, decision tree (DT) or clustering algorithms to initialize the population [14,6,15–17].
Furthermore, if variable selection is applied during the initialization and only the relevant variables are used to
form the fuzzy rules, EAs need to search the appropriate rules and parameters of fuzzy sets only for the
reduced set of variables. That clearly reduces the search space of EAs.

As illustrated above, many FC identification methods using EAs have been developed. They, however, have
some limitations, which are listed next. Some of the methods do not tune the fuzzy sets and require a priori
knowledge of the distribution of the fuzzy sets [9,1,18]. Some of the approaches initializes the population ran-
domly [10], which deteriorates the convergence. Also the variable selection in initialization phase is neglected
in many approaches [10,9,1,19,15,11,17]. Moreover some approaches use aggregated fitness functions
[20,16,6,7,21].

To the best of our knowledge, a method which initializes the population adequately (i.e. selects the relevant
variables, creates the relevant initial rules and partitions the input space adequately), tunes the membership
functions, and identifies a set of Pareto-optimal FCs has not been developed yet. This paper aims to fill that
gap.

In this paper the initial population is created in two phases. First a DT is created by C4.5 algorithm [22].
Because of the rectangular decision boundaries of crisp DTs, they can be overly complex. FCs, however, can
create non-axis parallel decision boundaries [23,24]. Therefore, DT is converted into an FC [6]. Because widely
distributed initial population improves the convergence of MOEAs [12,13], the rest of the population is cre-
ated by randomly replacing some parameters of the initial FC by random numbers, such that, the population
is widely distributed. DT initialization was previously applied, for example, in [6,7]. However, in those
approaches further optimization by EAs was performed using aggregated fitness functions and therefore a
set of Pareto-optimal FCs was not obtained.
NSGA-II algorithm [8] is applied to optimize the initial population and to find a set of Pareto-optimal FCs. It was successfully applied in [25,1] for the same purpose. However, in this paper NSGA-II is also applied to fine-tune the parameters of fuzzy sets, not only to find the appropriate rules and rule conditions. Furthermore, only the relevant variables, selected by C4.5 algorithm, are used to form fuzzy rules.

The rest of this paper is organized as follows. Section 2 briefly represents the theory behind multiobjective problems (MOPs) and NSGA-II algorithm. Furthermore, FCs are introduced and the criteria defining their fitness is presented. Section 3 represents the proposed FC identification method. It introduces the proposed two-step initialization method and presents the coding of the FC into a chromosome, such that, NSGA-II algorithm can be applied. In Section 4 performance of our method is studied on six benchmark data sets and the obtained results are compared to the results in the literature. The results show that by the proposed method a compact set of high quality solutions is obtained. Finally, Section 5 concludes the paper.

2. Preliminaries

In this section a brief introduction to the theory of multiobjective problems (MOPs) is given first. Then, NSGA-II [8], a popular multiobjective evolutionary algorithm (MOEA) applied in this paper, is briefly presented. After that, the basic theory of fuzzy classifiers (FCs) is given. Finally, the fitness function applied in this paper is defined.

2.1. Multiobjective problems

Let us assume a MOP with \( h \) objectives \( f_i, i = 1, \ldots, h \). Let \( \mathbf{s} \) be the decision vector and \( S \) the feasible region of the decision vector. That MOP can be formulated as:

\[
\text{Minimize } f_1(\mathbf{s}), f_2(\mathbf{s}), \ldots, f_h(\mathbf{s}) \text{ subject to } \mathbf{s} \in S.
\]  

(1)

It is often impossible to find a solution which simultaneously minimizes all \( h \) objectives. Hence, a set of widely spread trade-off solutions is often sought. A particular interest is on the non-dominated (Pareto optimal) decision vectors. A decision vector \( \mathbf{s}_1 \in S \) is Pareto optimal, if there does not exist a decision vector \( \mathbf{s}_2 \in S \), which fulfills the following conditions:

\[
\forall i, \quad f_i(\mathbf{s}_2) \leq f_i(\mathbf{s}_1) \quad \text{and} \quad \exists j, \quad f_j(\mathbf{s}_2) < f_j(\mathbf{s}_1).
\]  

(2)

If \( \mathbf{s}_2 \) meets the conditions in (2), it dominates \( \mathbf{s}_1 \). The Pareto-optimal set is formed of non-dominated solutions and their image under the objective functions is the Pareto front. [26,1].

2.2. Multiobjective evolutionary algorithms

MOEAs have been widely used to solve MOPs. Some of the application areas are, for example, the stochastic multiobjective environmental/economic dispatch problems [27] and scheduling of drilling operations [28]. Like mentioned earlier, they have also been used to design the fuzzy classifiers and function estimators.

NSGA-II [8] is a popular MOEA. It is a well-applicable algorithm, because it includes, for example, an efficient constraint-handling method, a fast non-dominated sorting procedure, an elitist approach and uses parameterless crowding distance measure to maintain the diversity of population. It is applied in this paper with polynomial mutation and simulated binary cross-over (SBX) [29] as genetic operators. The details of NSGA-II are not given in this paper but they can be found from [8]. Other good MOEAs are SPEA2 [30] and \( \epsilon \)-MOEA [31], just to mention a few.

2.3. Fuzzy classifiers

Fuzzy classification rules consist of fuzzy sets in the antecedent and a class label in the consequent. Let us denote the data set with \( D \) data points and \( n \) variables as \( \mathbf{Z} = [\mathbf{X} \ \mathbf{y}] \), where input matrix \( \mathbf{X} \) and output vector \( \mathbf{y} \) are given as:

\[
\mathbf{Z} = [\mathbf{X} \ \mathbf{y}]
\]
According to [6] fuzzy classification can be performed as follows:

\[ R_i: \ \text{If } x_1 \text{ is } A_{i,1} \ldots \text{ and } x_n \text{ is } A_{i,n} \text{ then } g_i, \ i = 1, \ldots, R, \]

where \( R \) is the number of rules, \( A_{i,j}, j = 1, \ldots, n \) is a membership function, \( g_i \in \{1, \ldots, C\} \) is the rule consequent and \( C \) is the number of different classes in data set. For each data point \( x_k \), the degree of fulfillment of a rule is computed as:

\[ \beta_i(x_k) = \prod_{j=1}^{n} A_{i,j}(x_{kj}). \]

The rule with the highest degree of fulfillment is declared as the winner rule (i.e. Winner takes all strategy). The output of the classifier is the rule consequent associated to that rule. There are also other types of fuzzy rules and \( t \)-norms which can be applied to reasoning [32] and the properties of fuzzy classifiers are discussed in detail in [33].

2.4. Fitness of a fuzzy classifier

Accuracy of FCs is measured by calculating the number of misclassifications. However, there is no generic way to measure the interpretability of FCs [34]. Often the interpretability is measured by calculating the number of rules and the total number of antecedents in the rules (total rule length) [18]. It was stated in [1], that the number of rules together with the total rule length can prevent overfitting. Consequently, it is beneficial to use both of those objectives. So in this paper, the objectives to be minimized are the number of misclassifications, the number of rules and the total rule length.

3. Proposed hybrid fuzzy classifier identification method

This section introduces the hybrid fuzzy classifier (FC) identification method, which is based on decision tree (DT) and multiobjective evolutionary algorithms (MOEA). When any standard MOEA is applied, the first step is the creation of the initial population. In contrast to many existing methods, the initial population is not created randomly or based on a priori knowledge, but by a two-step initialization method. First, an FC is identified using C4.5 algorithm. Then, to improve the convergence of EAs, the rest of the population is created by randomly replacing some parameters of that FC such a way, that the initial population is widely distributed. Finally, NSGA-II algorithm is applied to optimize the initial population and a set of non-dominated solutions is obtained. The proposed method is summarized in Fig. 1.

The rest of this section is organized as follows. First, FC initialization by C4.5 algorithm is discussed. Then, coding of an FC into a chromosome is presented and an illustrative example is given. Finally, it is introduced how the rest of the population is created in a way that the initial population is widely spread.

---

**Fig. 1.** Proposed hybrid fuzzy classifier identification method.
3.1. Initialization of FCs

First C4.5 algorithm is applied to create a decision tree (DT). C4.5 was selected as a part of initialization, because it is a top rated DT algorithm [35], it can select the relevant variables and partition the input space [6]. DT is then converted into an FC like shown in [6]. That can be done without decomposition error, if trapezoidal membership functions (MFs) are used, such that, they present the crisp decision boundaries of DT [6]. In this paper, however, the crisp decision boundaries are softened by using generalized bell (gbell) MFs:

\[
f(x; a, b, c) = \frac{1}{1 + \left| \frac{x - c}{a} \right| ^ b},
\]

where \( x \) is the data point, and \( a, b \) and \( c \) are the parameters of a gbell MF. The value of \( b \) defines the fuzziness of a MF. If it is set to a high value, say more than 100, then a MF is very close to a crisp function. Therefore, in this paper \( 0 < b < 10 \).

The value of \( a \) is restricted as:

\[
\max(0, a_{\text{initial}} * (1 - \alpha)) < a < a_{\text{initial}} * (1 + \alpha),
\]

where \( a_{\text{initial}} \) denotes the value of \( a \) when a DT is converted into an FC. Value of \( \alpha = 1/M_j \) defines how much parameters can vary around their initial values [14]. \( M_j \) stands for the maximum number of fuzzy sets assigned to a variable \( j \) and equals to the number of fuzzy sets in variable \( j \) in initial FC.

The value of \( c \), which defines the center of gbell MF, is restricted as:

\[
\max(c_{\text{initial}} - \alpha * \chi, \text{lbound}) < c < \min(c_{\text{initial}} + \alpha * \chi, \text{ubound}),
\]

where ‘ubound’ and ‘lbound’ are respectively the upper and lower bounds of a variable and \( \chi = \text{ubound} - \text{lbound} \) denotes the range of a variable.

There are also other reasons for applying gbell MFs instead of trapezoidal MFs. Gbell MFs may have better fit to the data [36] and they have three parameters in contrast to four parameters of trapezoidal MFs. Furthermore, the parameters of gbell MFs can be optimized independently, which is not the case when trapezoidal MFs are used. Therefore, standard mutation and cross-over operators of NSGA-II algorithm can be used without the need to make sure that, for example, parameter \( b \) is greater than parameter \( a \). The decomposition error caused by transformation of trapezoidal MFs into gbell MFs can be usually overcome by EA optimization [7].

3.2. Structure of a chromosome

Each individual (chromosome) contains an FC. Their structure is coded as a real coded vector including antecedents of the rules \( A \) and parameters of the fuzzy sets \( P \). Antecedent vector \( A \) is defined as:

\[
A = (A_{1,1}, A_{1,2}, \ldots, A_{1,n_s}, A_{2,1}, A_{2,2}, \ldots, A_{2,n_s}, \ldots, A_{R,1}, A_{R,2}, \ldots, A_{R,n_s}),
\]

where \( R \) denotes the number of rules in initial FC and \( n_s \) stands for the number of variables selected from \( n \) variables by C4.5 algorithm. Naturally \( n_s \leq n \), but usually \( n_s < n \). Since real coding of the variables is used for all parameters, the integers \( A_{i,j} = \{0, 1, \ldots, M_j\} \), indicating which membership function is used for variable \( j \) in rule \( i \), are coded as real coded values \( A_{i,j}^* \), which are rounded to the nearest integer when fitness evaluation is performed. Therefore, \( A_{i,j} = \text{round}(A_{i,j}^*) \), where \(-0.5 < A_{i,j}^* < M_j + 0.5 \).

During MOEA optimization, number of rules, rule conditions and variables can be decreased. If variable \( j \) is not used in rule \( i \), then \( A_{i,j} = 0 \). If rule \( i \) is not used in an FC, then \( \forall j, A_{i,j} = 0 \). If variable \( j \) is not used in an FC, then \( \forall i, A_{i,j} = 0 \). In this paper it is required that each chromosome has at least one antecedent and one rule (i.e. \( \exists i, \exists j, A_{i,j} \neq 0 \)).

Parameter vector \( P \) is given as:

\[
P = (P_1,1, P_1,2, \ldots, P_{1,\beta}, P_{2,1}, P_{2,2}, \ldots, P_{2,\beta}, \ldots, P_{\gamma,1}, P_{\gamma,2}, \ldots, P_{\gamma,\beta}),
\]

where \( \gamma \) is the number of parameters used to define a membership function and \( \beta = \sum_{j=1}^{n_s} M_j \) is the total number of fuzzy sets in initial FC. In this paper Gbell membership functions are used, so \( \gamma = 3 \).
Consequent part of the fuzzy rule $g = (g_1, \ldots, g_R)$ is not included in an individual. It is static and created in initialization phase by C4.5 algorithm. So, NSGA-II is used to select rules, rule antecedents and parameters of membership functions for the pre-specified class labels. The total number of parameters $\theta$ to be optimized by NSGA-II algorithm is therefore given as:

$$\theta = R \times n_s + \gamma \times \beta.$$ (9)

Each parameter is restricted with lower and upper bounds defined in current and previous subsections. Therefore the number of constrains is $2 \times \theta$.

### 3.3. Coding of a chromosome: an example

Let us consider a classification problem with 5 classes and 4 variables. Let us assume that an FC with 5 rules and 5 fuzzy sets has been created by transforming a DT into an FC. C4.5 algorithm has selected 2 variables, $x_1$ and $x_2$, assigned 3 fuzzy sets to variable $x_1$ and 2 fuzzy sets to variable $x_2$. The obtained rules with total rule length of 9 are the following:

- **Rule 1**: If $x_1$ is 1 and $x_2$ is 1 then Class is 5
- **Rule 2**: If $x_1$ is 1 and $x_2$ is 2 then Class is 4
- **Rule 3**: If $x_1$ is 2 and $x_2$ is 1 then Class is 3
- **Rule 4**: If $x_1$ is 2 and $x_2$ is 2 then Class is 2
- **Rule 5**: If $x_1$ is 3 then Class is 1

Coding of the antecedent part would be then:

$$A = \begin{pmatrix}
1, 1, 1, 2, 2, 1, 2, 2, 3, 0
\end{pmatrix}_{\text{Rule}_1 \text{ Rule}_2 \text{ Rule}_3 \text{ Rule}_4 \text{ Rule}_5}.$$

Coding of the five membership functions would be:

$$P = \begin{pmatrix}
P_{1,1}, P_{1,2}, P_{1,3}, P_{1,4}, P_{1,5}, P_{2,1}, P_{2,2}, P_{2,3}, P_{2,4}, P_{2,5}, P_{3,1}, P_{3,2}, P_{3,3}, P_{3,4}, P_{3,5}
\end{pmatrix}_{\text{Gbell parameter } a \text{ Gbell parameter } b \text{ Gbell parameter } c}.$$

The rule consequents, which are the same for all individuals and not included in an individual are:

$$g = \begin{pmatrix}
5, 4, 3, 2, 1
\end{pmatrix}_{\text{Rule}_1 \text{ Rule}_2 \text{ Rule}_3 \text{ Rule}_4 \text{ Rule}_5}.$$

### 3.4. Initializing the rest of the population

The rest $N - 1$ chromosomes, where $N$ is the population size, are created by randomly replacing some parameters of the FC created by C4.5 algorithm in Section 3.1. The replacement algorithm creates a set of widely distributed chromosomes and it is given next:

**Repeat for** $I = 1, \ldots, N - 1$, where $I$ is the chromosome iterator.

**Step 1**: Calculate the number of parameters to be replaced $m$ as follows:

$$m = \text{round} \left( \frac{I}{N - 1} \times \theta \right),$$ (10)

where ‘round’ stands for the operator rounding the result to the nearest integer.

**Step 2**: Choose randomly $m$ parameters out of $\theta$. 
Step 3: Replace them by randomly generating \( m \) parameters between their corresponding limits, defined in Sections 3.1 and 3.2.

End for

So the algorithm above generates widely distributed chromosomes, which all share the same structure, defined in Section 3.2. Some of the chromosomes are either very similar or very different to the chromosome generated by C4.5 algorithm, whereas some are between those extremes. That is important, because it speeds up the convergence of EAs to the correct Pareto front [13].

4. Experiments

In this section performance of the proposed identification framework is validated. First, the experimental setup used in this paper is described. Then, as illustrative examples, well-known Wine and Sonar classification data sets are studied. Finally, six benchmark data sets from UCI Machine Learning Repository [37] are studied and a rigorous comparison of our results to the results presented in [1] is performed.

4.1. Experimental setup

Six benchmark data sets, Wisconsin breast cancer (Wisc), Pima Indians diabetes (Pima), Glass, Cleveland heart disease (Cleve), Sonar, and Wine were studied in this paper. These data sets represent problems with different number of classes, variables and data points (see Table 1) and they were also studied in [1]. Wisconsin breast cancer and Cleveland heart disease data sets contained data points with missing values. Those data points were removed.

In [1] a multiobjective fuzzy genetics-based machine learning (GBML) algorithm based on NSGA-II algorithm was applied to obtain a set of non-dominated fuzzy rule-based classifiers. The parameters of fuzzy sets were pre-specified by partitioning each input variable with 14 fuzzy sets and with a “don’t care” value. So the problem was to specify the appropriate number of rules and to select the antecedents to those rules from those aforementioned 15 fuzzy sets. Variable selection was not applied before executing GBML algorithm, so for each rule \( n \) antecedents need to be specified. However, GBML algorithm can remove variables by assigning “don’t care” values for certain variables in all rules.

Three formulations of multiobjective optimization problems (MOPs) were applied in [1]. MOP-1 was used to maximize the accuracy and to minimize the number of rules. MOP-2 was applied to maximize the accuracy and to minimize the total number of conditions in rules. MOP-3 was used to maximize the accuracy, minimize the total number of conditions in rules and to minimize the number of rules. So MOP-3 uses the same fitness function as in this paper.

In [1], the number of generations and population size were set to 5000 and 200, respectively. Like illustrated in [1], the search space is enormous, even when the parameters of fuzzy sets are pre-specified. In our case, when the parameters of fuzzy sets need to optimized as well, it is beneficial to use larger population size [38]. However to perform a fair comparison, the number of fitness evaluations was limited to 1 000 000 \(^1\) (i.e. \( 5000 \times 200 \)). So in this paper the number of generations and population size were both set to 1000. The applied parameters for NSGA-II algorithm are shown in Table 2. The same cross-over and mutation probabilities and distribution indexes were also applied in [8]. For C4.5 algorithm, the pruning confidence value was set to 5 in order to reduce the complexity of the initial FCs. The rest of the parameters were kept as their default values defined in [22].

4.2. Illustrative examples

The purpose of this subsection is to shed light on the proposed method by identifying FCs for Wine and Sonar classification data sets (see also Table 1). Those data sets were selected as examples, because they

\(^1\) It was, however, illustrated in [1], that better results can be obtained by increasing the number of fitness evaluations.
represent problems with different complexity, that is, Wine data have a moderate number of variables (13 variables) and Sonar data have a high number of variables (60 variables).

4.2.1. An illustrative example 1: wine classification

The problem is to classify 3 types of wines based on 13 variables, labeled here as $x_1, x_2, \ldots, x_{13}$. There are total of 178 datapoints in the data set. To perform the experiments, the data set was randomly divided into training and testing set. Training set consisted of 80% of datapoints, that is, 142 datapoints and testing set the rest 20% of datapoints (36 datapoints).

![Decision Tree](image-url)

Fig. 2. Wine data: the obtained decision tree.
C4.5 algorithm was run and a decision tree (DT) was obtained (see Fig. 2). It was converted into an FC consisting of 7 rules and 21 rule conditions. The initial FC used six variables, $x_4$, $x_7$, $x_{10}$, $x_{11}$, $x_{12}$, and $x_{13}$, selected by C4.5. The fuzzy sets of the initial FC are shown in Fig. 3. Since each of the 6 variables are partitioned with 2 fuzzy sets, they can be labeled, for example, as small and large. Therefore, seven rules, generated by collecting all the conditions on the way from the root of the tree to each of the seven leaves of DT [6] can be expressed as:

\begin{align*}
\text{If } x_{13} \text{ is large and } x_{11} \text{ is small then Class is 3} \\
\text{If } x_{13} \text{ is large and } x_{11} \text{ is large and } x_{10} \text{ is large then Class is 1} \\
\text{If } x_{13} \text{ is large and } x_{11} \text{ is large and } x_{10} \text{ is small then Class is 2} \\
\text{If } x_{13} \text{ is small and } x_{12} \text{ is large then Class is 2} \\
\text{If } x_{13} \text{ is small and } x_{12} \text{ is small and } x_{4} \text{ is small then Class is 2} \\
\text{If } x_{13} \text{ is small and } x_{12} \text{ is small and } x_{4} \text{ is large and } x_{7} \text{ is large then Class is 2} \\
\text{If } x_{13} \text{ is small and } x_{12} \text{ is small and } x_{4} \text{ is large and } x_{7} \text{ is small then Class is 3}
\end{align*}

After that, the rest of the population was created by randomly modifying some parameters of the initial FC, like illustrated in Subsection 3.4. The initial population is shown in Fig. 4.

Then, MOEA optimization was performed and a set of Pareto-optimal FCs was obtained. It can be seen from Fig. 4 that complexity of FCs was highly reduced due to MOEA optimization. That, however, did not deteriorate the accuracy. From the set of Pareto-optimal FCs, an FC can be selected based on the
preferences. For example, an FC with 4 rules and 6 rule conditions can be selected. Its fuzzy sets are shown in Fig. 5 and its rules are the following:

- If \( x_7 \) is small then Class is 3
- If \( x_{10} \) is small then Class is 2
- If \( x_7 \) is large and \( x_{13} \) is small then Class is 2
- If \( x_{10} \) is large and \( x_{13} \) is large then Class is 1

The properties of the selected FC along with properties of DT and initial FC are shown in Table 3. It is seen that the selected FC is highly accurate, yet it is the most interpretable solution. From that Table it can be noticed that the training error of initial FC is slightly worse than training error of DT. That is due to the decomposition error caused when DT was converted into FC.

4.2.2. An illustrative example 2: sonar classification

Sonar data with 60 variables is studied in order to emphasize the variable selection capability of the proposed method. Sonar data consist of 111 and 97 patterns obtained by bouncing off the sonar signals from metal cylinders and rocks, respectively [39]. The problem is to distinguish between those signals based on 60 variables, named here \( x_1, x_2, \ldots, x_{60} \).

The experiments were carried out exactly the same manner as in the previous example, but for the sake of brevity, DT and the fuzzy sets of initial and final FCs are not shown here. C4.5 selected 11 variables, namely \( x_8, x_{11}, x_{13}, x_{36}, x_{45}, x_{51}, x_{53}, x_{54}, x_{59}, x_{60} \). The initial population and the final population after MOEA optimization are shown in Fig. 6. It is seen that even the dimensionality of the problem is high, the complexity
of the initial FC is still moderate, consisting of 12 rules and 49 rule conditions. In Table 4 properties of four FCs of final population are shown along with the initial FC and DT. Selected FC 1 is the most complex FC of the final population and it consists of 10 rules and 26 rule conditions in contrast to 12 rules and 49 rule conditions of the initial FC. Yet, selected FC 1 is more accurate than more complex initial FC. It is also noticed from Table 4 that the best testing accuracy is obtained when the number of rules and rule conditions are 3 and 4, respectively.

4.3. Results comparison

For each of the six data sets, described in Table 1, a 10-fold cross-validation (10-CV) [40,35] was performed 10 times (i.e. 10 × 10-CV). So the total number of runs for each data set was 100. A different random seed was used for each of the ten 10-CV runs. Since NSGA-II algorithm was applied, a set of non-dominated solutions was obtained for each run. Usually those sets are not identical. They may contain solutions with different structures (i.e. the number of rules and number of rule conditions) and the number of different solutions in

Table 3
Wine data: result comparison for decision tree, initial FC and selected FC

<table>
<thead>
<tr>
<th>Method</th>
<th>Train error rate</th>
<th>Test error rate</th>
<th>Rules</th>
<th>Total rule length</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4.5</td>
<td>0.0211</td>
<td>0.0278</td>
<td>7</td>
<td>21</td>
<td>6</td>
</tr>
<tr>
<td>Initial FC</td>
<td>0.0282</td>
<td>0.0278</td>
<td>7</td>
<td>21</td>
<td>6</td>
</tr>
<tr>
<td>Selected FC</td>
<td>0.0141</td>
<td>0.0000</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

Fig. 5. Wine data: obtained fuzzy sets for the selected FC after MOEA optimization.
a set may vary as well. In [1] only the solutions which were present in at least in 51 out of 100 runs were repre-
represented to have reliable results and we did that as well in order to perform a fair comparison. So the average 
error rates on test set over the number of runs the solution was present were calculated. In this paper, there 
may exist several solutions with the same structure and training error, but with different test error. That is 
possible because the parameters of fuzzy sets are not fixed. In those cases, the average of those test errors 
was selected to represent the solution for that run.

The solutions are presented in Figs. 7–12 along with the results of [1].² It is noted that all of our solutions 
for Wisconsin breast cancer, Glass and Wine data sets are not dominated by MOP-1, MOP-2 or MOP-3.

² The exact values to reconstruct the figures were kindly provided by Hisao Ishibuchi and Yusuke Nojima, the authors of that paper.
Moreover, most of our solutions dominate some solutions of MOP-1, MOP-2 and MOP-3. For Pima Indians diabetes data, some of our solutions are dominated by MOP-3, but one of our solutions also dominates the solutions of MOP-1, MOP-2 and MOP-3. Our results for sonar data are quite similar to MOP-2 and MOP-3. Some of our solutions for Cleveland heart disease data clearly dominate the solutions of MOP-1, MOP-2 and MOP-3. However, one of our solutions with 1 rule and 1 rule condition is dominated by MOP-3.

Average best error rates for test and training sets over the 100 runs were calculated and presented in Tables 5 and 6. For comparison, the results of SOP-1, SOP-2 and SOP-3 [1], the single-objective versions of MOP-1,
MOP-2 and MOP-3, respectively, are included in those Tables. Moreover, the best results of six C4.5 variants, representing different splitting strategies and evaluation functions [41], are included in Table 6. Their performances were evaluated by $10 \times 10$-CV, which is the same experimental setup as in this paper.

It is noted from Table 5 that by means of our method the lowest training error rates were obtained for four out of six data sets. Furthermore, it can be seen from Table 6 that the lowest testing error rates for five
out of six data sets were also obtained by our method. That indicates good generalization capabilities of our method.

In Table 7 the average number of the obtained non-dominated solutions is presented. It is noted that generally our method obtained less non-dominated solutions than MOP-3, which has the same fitness function as our method. Only for one out of six data sets our method obtained more non-dominated solutions than MOP-3. That was due to the initialization algorithm used in this paper. Because C4.5 algorithm selected the relevant variables and created moderate number of rules and fuzzy sets, the number of possible solutions was reduced. However, as Tables 5 and 6 and Figs. 7–12 indicated, the quality of the obtained solutions was high.

A reader may wonder, that the number of obtained non-dominated FCs in Table 7 is higher than in Figs. 7–12. That is due to the experimental setup, which requires that an FC with a certain structure (i.e. certain number of rules and certain total rule length) must be present at least in 51 out of 100 runs. FCs which consist of few rules have less conceivable structures and therefore they are more likely to be presented in those aforementioned figures. To illustrate the effect of the experimental setup, Fig. 10 is constructed again, such that, one of the hundred runs is selected and the non-dominated solutions for that run are shown in Fig. 13. By comparing Figs. 10 and 13, it is seen that FCs with more than 5 rules are present in Fig. 13, but not in Fig. 10. A clear trade-off structure for train set is seen in Fig. 13. That is, however, not the case when test set is considered. Some of the more complex solutions show poor generalization capabilities due to the overfitting. That confirms again, that the number of rules together with total rule length can prevent overfitting, like illustrated in [1].

### Table 5

<table>
<thead>
<tr>
<th>Data</th>
<th>This paper</th>
<th>MOP-1</th>
<th>MOP-2</th>
<th>MOP-3</th>
<th>SOP-1</th>
<th>SOP-2</th>
<th>SOP-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>21.48</td>
<td>13.98</td>
<td>16.94</td>
<td>27.09</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sonar</td>
<td>16.84</td>
<td>10.01</td>
<td>16.49</td>
<td>20.47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wine</td>
<td>23.35</td>
<td>11.56</td>
<td>11.45</td>
<td>11.81</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cleveland</td>
<td>25.11</td>
<td>27.08</td>
<td>25.94</td>
<td>25.72</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wisconsin</td>
<td>1.94</td>
<td>1.59</td>
<td>1.71</td>
<td>1.08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pima</td>
<td>16.17</td>
<td>19.48</td>
<td>19.79</td>
<td>19.59</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 6

<table>
<thead>
<tr>
<th>Data</th>
<th>This paper</th>
<th>MOP-1</th>
<th>MOP-2</th>
<th>MOP-3</th>
<th>SOP-1</th>
<th>SOP-2</th>
<th>SOP-3</th>
<th>C4.5 in [41]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>24.05</td>
<td>35.55</td>
<td>33.93</td>
<td>34.05</td>
<td>35.76</td>
<td>39.21</td>
<td>38.36</td>
<td>27.3</td>
</tr>
<tr>
<td>Sonar</td>
<td>16.73</td>
<td>23.18</td>
<td>17.32</td>
<td>17.51</td>
<td>24.04</td>
<td>23.47</td>
<td>24.29</td>
<td>24.6</td>
</tr>
<tr>
<td>Wine</td>
<td>2.98</td>
<td>3.99</td>
<td>3.65</td>
<td>3.04</td>
<td>7.30</td>
<td>6.49</td>
<td>6.52</td>
<td>5.6</td>
</tr>
<tr>
<td>Cleveland</td>
<td>38.22</td>
<td>42.57</td>
<td>42.85</td>
<td>42.64</td>
<td>44.83</td>
<td>45.80</td>
<td>45.44</td>
<td>46.3</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>2.95</td>
<td>2.93</td>
<td>2.74</td>
<td>2.66</td>
<td>3.88</td>
<td>3.69</td>
<td>3.56</td>
<td>5.1</td>
</tr>
<tr>
<td>Pima</td>
<td>21.78</td>
<td>23.27</td>
<td>22.32</td>
<td>21.80</td>
<td>25.26</td>
<td>25.00</td>
<td>24.20</td>
<td>25.0</td>
</tr>
</tbody>
</table>

### Table 7

<table>
<thead>
<tr>
<th>Data</th>
<th>This paper</th>
<th>MOP-1</th>
<th>MOP-2</th>
<th>MOP-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>21.48</td>
<td>13.98</td>
<td>16.49</td>
<td>27.09</td>
</tr>
<tr>
<td>Sonar</td>
<td>16.84</td>
<td>10.01</td>
<td>20.47</td>
<td>17.66</td>
</tr>
<tr>
<td>Wine</td>
<td>6.03</td>
<td>11.45</td>
<td>9.96</td>
<td>11.81</td>
</tr>
<tr>
<td>Cleveland</td>
<td>23.35</td>
<td>11.56</td>
<td>22.17</td>
<td>18.59</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>5.64</td>
<td>12.09</td>
<td>13.32</td>
<td>12.25</td>
</tr>
<tr>
<td>Pima</td>
<td>13.52</td>
<td>9.71</td>
<td>15.80</td>
<td>17.06</td>
</tr>
</tbody>
</table>
5. Conclusions

This paper proposed a hybrid method for identification of Pareto-optimal fuzzy classifiers (FCs). In contrast to many existing methods, the initial population for the multiobjective evolutionary algorithms (MOEAs) was neither created randomly nor a priori knowledge was required. Instead of those techniques, a two-step initialization method was applied. First, an FC was obtained by transformation of a decision tree (DT) into an FC. Therefore, no a priori knowledge of the relevant variables, number of fuzzy sets or distribution of fuzzy sets was required. Then, the rest of the population was created by randomly replacing some parameters of that FC, in a way that the population was widely spread. That improved the convergence of MOEAs into the correct Pareto front.

FCs were coded in a way that a popular MOEA, named NSGA-II, could be used to select rules, rule antecedents and parameters of membership functions for the class labels specified by DT algorithm in initialization phase. Because the parameters of fuzzy sets were not static, it enabled us to approximate the distribution of data more accurately.

Number of misclassifications, number of rules and total rule length were used as objectives to be optimized. In the future, it can be considered, whether it is beneficial to use different objectives. For example, number of misclassifications could be replaced with the area under the receiver operating characteristic curve (AUC), which is useful when class distributions and misclassification costs are unknown [10]. Also the number of membership functions could be a object to be minimized. Those modifications to the fitness function can be easily done without affecting any other part of the proposed method. In the future it can also be considered whether MOEA should be used to modify the class labels of consequents as well. That may be useful in cases when genetic operators significantly modify the antecedents of the rules and therefore the class labels specified in initialization phase may not be adequate anymore.

Fig. 13. Cleveland heart disease data: the results of one run, selected out of 100 runs. A clear trade-off structure is seen for train set, but that is not so clear for test set due to the overfitting.
The validity of the proposed method was confirmed through six well-known benchmark data sets from the literature. We compared our results to another FC identification method by Ishibuchi and Nojima [1], which also utilized NSGA-II algorithm. The number of obtained Pareto-optimal solutions by our method was usually lower than in the comparative study. That was due to the initialization algorithm used in this paper. Because C4.5 algorithm selected the relevant variables and created moderate number of rules and fuzzy sets, the number of possible Pareto-optimal solutions was reduced. However, the variable selection also reduced the computational costs significantly. Furthermore, the quality of the obtained solutions was higher than in the comparative study; in five out of six data sets, we obtained more accurate solutions. Moreover, in three data sets, none of our solutions were dominated by the solutions of the comparative study and some of our solutions dominated the solutions of that study.

Acknowledgement

The authors want to express their gratitude to Hisao Ishibuchi and Yusuke Nojima for providing their results for results comparison. In addition, the comments of anonymous reviewers are greatly appreciated.

References


Publication P2


© 2008 Elsevier Ltd. Reprinted with permission.
Developing a bioaerosol detector using hybrid genetic fuzzy systems

Pietari Pulkkinen\textsuperscript{a,*}, Jarmo Hytönen\textsuperscript{b}, Hannu Koivisto\textsuperscript{a}

\textsuperscript{a}Department of Automation Science and Engineering, Tampere University of Technology, P.O. Box 692, FIN-33101 Tampere, Finland
\textsuperscript{b}Dekati Ltd., Osuusmyllykatu 13, FIN-33700 Tampere, Finland

Received 20 November 2007; accepted 9 January 2008
Available online 3 April 2008

Abstract

The aim of this work is to develop a model, which works as a reasoning mechanism in a bioaerosol detector. Ability to distinguish between safe and harmful aerosols is one of its main requirements. Instead of commonly used misclassification rate as a metric of accuracy, true positive (TP) and false positive (FP) rates are used because of the uneven misclassification costs and class distributions of the collected data. Interpretability of the model builds up the confidence for the developed model and enables its adjustment in cases when bioaerosol detector is further developed. Thus, it is another crucial requirement for the model. Clearly, the objectives are contradicting and therefore multiobjective evolutionary algorithms (MOEAs) are applied to find tradeoff models. Fuzzy classifiers (FCs) are selected as a model type because their linguistic rules are intuitive to human beings. FCs are identified by hybrid genetic fuzzy system (GFS) which initializes the population adequately using decision trees (DTs) and simplification operations. During MOEA optimization transparency of fuzzy partition is used as a metric of interpretability and TP and FP rates as metrics of accuracy. Heuristic rule and rule condition removal is applied to offspring population in order to keep the rule base consistent. The identified FCs are highly comprehensible yet accurate and their linguistic rules provide valuable insights for further development of bioaerosol detector.

Keywords: Fuzzy classifiers (FCs); Multiobjective evolutionary algorithms (MOEAs); Bioaerosol detector; Genetic fuzzy systems (GFS); Transparency

1. Introduction

Distinguishing between harmful and safe aerosols according to several measurement signals is the main purpose of a bioaerosol detector. Commonly, the reasoning is based on UV-fluorescent and the size of the particles (Sivaprakasam et al., 2004; Janka et al., 2007) and the first bioaerosol detector based on those signals was proposed in 1997 (Hairston et al., 1997). There are, however, some aerosols (e.g. soot from diesel engines) which may cause fluorescent response. Therefore those aforementioned signals are not sufficient for reliable detection. Therefore, in Janka et al. (2007) the UV-fluorescence detection optics were combined with a special background-aerosol detector system.

The purpose of this work is to develop a model to be used as an automatic reasoning method in the bioaerosol detector developed in Janka et al. (2007). The model must infer based on the available real-time measurement signals whether there are harmful or safe aerosols in the air. The further analysis of bioaerosols can be done after an alarm is issued online. The detector contains a dry filter, which can be removed for off-line analysis in a laboratory. That can be an expensive and time consuming operation (Hairston et al., 1997). Thus, it is important to minimize the number of false alarms. However, an alarm should always be raised whenever harmful particles are in the air.

The collected real-world bioaerosol data are highly imbalanced; there are much more data points representing harmless particles than harmful particles. Therefore, the commonly applied misclassification rate is not a suitable accuracy metric. Instead, true positive (TP) rate and false positive (FP) rate are used and the goal is to maximize TP rate and to minimize FP rate.
In addition to the requirements above, it is very important to understand the reasoning of the developed model. That builds up the confidence that the developed model actually works reasonably, which in real-world problems, is one of the requirements for the model to meet (Elder and Pregibon, 1996). Otherwise it may not be applied in practice due to lack of confidence in it. Furthermore, interpretability of the model can reveal dysfunction of the process/system and therefore aid in its further development.

There are several machine learning methods, such as, neural networks or support vector machines, which can lead to accurate results. However, due to their black-box nature, their inference is not easily comprehensive. Decision trees (DTs), for example C4.5 (Quinlan, 1993) the successor of ID3 (Quinlan, 1986), can be both accurate and interpretable. Moreover, they can automatically select relevant variables (Abonyi et al., 2003). They may, however, result in more complex trees than necessary due to their axes-parallel partition (Abonyi et al., 2003) and crispness, especially when applied to noisy data. That can be tackled by using fuzzy classifiers (FCs), which can deal with uncertainties in data. FCs, if identified adequately, can be highly interpretable due to their if–then rules, because that is the intuitive way to deduce for human beings. Furthermore FCs can be very accurate.

Recently, more FCs are identified by using evolutionary algorithms (EAs), because of their good learning capabilities for complex problems. Those approaches are often called genetic fuzzy systems (GFS) (Cordón et al., 2004). They may have multiple objectives, for example number of rules, total number of conditions and misclassification rate (Ishibuchi et al., 2001), which are to be minimized simultaneously. GFS can converge into a single solution when aggregated fitness function is used or into a set of Pareto-optimal solutions when multiobjective evolutionary algorithms (MOEAs) are used.

Commonly the initial population for GFS is created randomly or manually (Setzkorn and Paton, 2005; Ishibuchi et al., 2006; Gómez-Skarmeta et al., 1998), while better convergence due to reduction of the search space is obtained by adequate initialization (Haubelt et al., 2005; Poles et al., 2006). That can be done, for example, by using DTs or clustering algorithms and transforming DTs or clusters into FCs (Roubos and Setnes, 2001; Abonyi et al., 2003; Pulkkinen and Koivisto, in press).

A hybrid GFS is applied in this work. It initializes the population using DT algorithm and simplification operations (merging of fuzzy sets and reducing the number of rules). Then, the initial population is further optimized by MOEA. The hybrid GFS is a further development of our previous work (Pulkkinen and Koivisto, in press) and its objectives are transparency of fuzzy partition and TP and FP rates. When transparency of fuzzy partition is used as an objective, intuitive linguistic values for linguistic variables are obtained. Furthermore, the proposed GFS uses heuristic rule and rule condition reduction to improve the convergence of MOEAs and to avoid inconsistencies in rule base. By means of the proposed GFS, a widely distributed set of Pareto-optimal FCs, presenting the tradeoff between the objectives, was obtained.

Some promising preliminary results were reported in our former study (Pulkkinen et al., 2008) and in this paper a deeper analysis of them is performed. It is especially shown how the interpretable FCs can be used for further development of bioaerosol detector, which is currently under development. Based on an analysis of an FC, selected from the Pareto-optimal set of solutions, it was noticed that some of the false alarms were actually caused by dysfunction of bioaerosol detector. Therefore, clear guidelines for its further development were given and put into practice. Furthermore, we also validate the usefulness of the heuristic rule and rule conditions removal and simplification operators and conclude that those operators significantly reduce the computational costs and improve the quality of solutions.

The rest of this paper is organized as follows. Section 2 gives some insights to the constructed bioaerosol detector. Furthermore, collection of the data from the field and laboratory for teaching the model is discussed. Then, Section 3 describes the proposed hybrid GFS and discusses the fitness of FCs. After that in Section 4 the results of the identified FCs are presented followed by comparison of FCs identified by the proposed GFS with and without heuristic rule and rule condition removal and simplification operators. Moreover, the usage of interpretable FCs for further development of bioaerosol detector is described. Finally, Section 5 draws the conclusions.

2. Bioaerosol detector

This section briefly presents the applied bioaerosol detector. A reader is suggested to refer to Janka et al. (2007) for more information. A schematic diagram of bioaerosol detector is illustrated in Fig. 1. To reduce the false alarms caused by non-bioaerosols, it includes not only UV-fluorescence and particle size signals but also a special background-aerosol detector. That makes it different from the conventional bioaerosol detectors (Janka et al., 2007).

The function of bioaerosol detector is described as follows. First, particles go through a size selective sampling to a concentrator. In order to prevent the pollens to enter, the cut-off size for this inlet is selected as <7μm. The concentrator then gains the larger particles (>2μm) by factor >500. The concentrated larger particles then go to the optical measurement unit (also called primary unit) and their UV-fluorescence and elastic scatter are measured. The rest of the particles (i.e. the smaller ones <2μm) go to the secondary unit. Because the particles from combustion processes (e.g. exhaust gases from diesel engine) are of this size, the secondary unit is in key role to prevent false alarms (Janka et al., 2007).
When an alarm is raised, the sample collection is automatically started by channeling the aerosol flow to a dry filter. Then, the filter is removed and brought to the laboratory for further investigation (Janka et al., 2007).

The device has been constructed in the form of a suitcase (see Fig. 2). Its weight is currently 48 kg and its dimensions are:

- Height: 950 mm.
- Width: 526 mm.
- Depth: 310 mm.

2.1. Collecting and preprocessing of the data

Obtaining data which are reliable, that is, represent the true characteristics of the modeled object or system is a challenging task. It was stated in Liu et al. (1999), that up to 70% of project’s effort can be spend on this phase. In this work data were collected both from the field and from the laboratory. The field measurement campaign was performed at Umeå trial field in Sweden during early autumn 2006. The laboratory measurements were performed at Finnish Defense Forces Technical Research Center at the end of October 2006.

One of the major challenges for preparing the data for supervised learning algorithms is to assign each input data point to adequate class label. In this work there are only two classes, namely alarm and normal. Of course during the measurement period, it is exactly known when particles are spread. Furthermore, it is known whether the distributable particles are harmful or safe. However, there are many challenges in assigning the class label for the

Fig. 1. Schemata of the applied bioaerosol detector.

Fig. 2. The constructed bioaerosol detector in a suitcase like casing.
data, especially when the field measurements are considered. Some of them are listed next:

- It is not trivial to estimate when particles have spread from distribution point to the measurement point. That is dependent, for example, on the direction and speed of the wind.
- There can be some disturbances, for example, dust caused by walking people and smoke caused by cars or cigarettes.

In this work, the output was labeled based on the field log and expert knowledge. It was roughly known when the particles should reach the bioaerosol detector. Then, when a rise in a certain measurement value occurred, it was marked as the point particles had reached the bioaerosol detector. If those particles were harmful, the output was marked as alarm; otherwise it was marked as normal. Then when the value of that measurement descended, it was marked as the point the spreading of the particles was finished. Before and after the spreading period the output was marked as normal.

However, between the start and the end of the spreading period the direction of the wind may change. Therefore, between the start and the end of the spreading, there may be periods during which no particles are reached to the bioaerosol detector. If those data points are marked as alarm, the data become inconsistent. Thus, a threshold for the aforementioned measurement was defined based on expert knowledge. Hence, even if harmful particles are spread, but the threshold is not exceeded, those data points are still marked as normal. From the data points not exceeding the threshold, only a presentable subset was chosen in order to reduce the computational costs of fitness evaluations.

2.2. Anomalies in the data

Anomalies in data are always expected whenever large amounts of real-world data are collected. Furthermore, the bioaerosol detector is currently in prototype phase and many changes and improvements are expected in the near future. Because of that, it is natural that the collected data contain anomalies and extensive noise, which will not be the case anymore when the system is more established.

The collected data have some anomalies in absolute values of the measurements; there are changes in the values of the measurements when there is nothing spread in the air. Naturally those values which are called zero values in this work, should always be approximately the same. However, quite significant variations in zero values were present. Moreover, one of the measurements was very sensitive to temperature and caused trends in the data.

That makes the development of accurate models a very difficult task. However, it is important for the continuation of the project to show that the proposed bioaerosol detector has potential for distinguishing between harmful and safe particles. Keeping those factors in mind, it was not meaningful to clean the data from all anomalies and try to make the collected data idealistic. Anyway, when the bioaerosol detector is evaluated online, the data may currently have some anomalies. Nevertheless, the model should still work reasonably. Therefore, only the data points with most significant errors, for example a negative measurement value for a measurement which should always be positive or very strong trends, were removed. Also some zero values were slightly modified.

3. Proposed hybrid GFS

This section introduces the proposed hybrid GFS, which is a refinement of our earlier work (Pulkkinen and Koivisto, in press). It combines the advances of DTs, multiobjective evolutionary algorithms (MOEAs), similarity based merging of fuzzy sets, and heuristic rule and rule conditions removal to identify a set of Pareto-optimal FCs.

The identification procedure is presented in Fig. 3. It is started with initialization of an FC with a crisp DT algorithm. That is clearly a better starting point for further optimization than commonly applied random initialization. However, due to crispness of DT and the noise in real-world data, this FC is overly complex and can be simplified (Abonyi et al., 2003). Therefore, the initial FC goes through merging of similar fuzzy sets, which may lead to similar (Setnes et al., 1998) or inconsistent rules. Those rules are heuristically removed in order to improve the convergence of MOEAs and to reduce computational costs of fitness evaluations.

After that, the rest of the population is created by modifying the simplified FC, such that, the initial population is widely spread. That is beneficial to the convergence of MOEAs (Poles et al., 2006; Haubelt et al., 2005). Finally, a MOEA is applied to find a set of widely spread Pareto-optimal FCs. During MOEA optimization the offspring population goes through rules and rule conditions reduction in order to prevent the rule base having inconsistent rules.

As a DT algorithm, C4.5 (Quinlan, 1993) is applied. Its advances include selection of input variables and partition of input space with non-fixed number of hyper-rectangles (Abonyi et al., 2003; Pulkkinen and Koivisto, 2007). It is a well known and widely used algorithm and therefore no details are given in this paper. An interested reader may refer to Quinlan (1993) and Abonyi et al. (2003) for further details.

As a MOEA component, NSGA-II (Deb et al., 2002), a popular and commonly applied MOEA is used. Its strengths include efficient method for constraint-handling, fast non-dominated sorting procedure, and parameterless crowding distance measure for maintaining diversity of population. Its details are not presented in this paper, since it is well documented in Deb et al. (2002).

The rest of this section is organized as follows. First, FCs and multiobjective problems (MOPs) are briefly presented.
Then, the hybrid GFS is described in detail. After that, metrics for accuracy and interpretability are discussed. Finally, the fitness function applied in this paper is presented. In this section the same notations as in Pulkkinen and Koivisto (in press) are used.

### 3.1. Fuzzy classifiers

A fuzzy classification rule consists of fuzzy sets in the antecedent and a class label in the consequent. Let the data set with \( D \) data points and \( n \) variables be denoted as \( \mathbf{Z} = [\mathbf{X} \mathbf{y}] \), where \( \mathbf{X} \) is \( D \times n \) input matrix and \( \mathbf{y} \) is \( D \times 1 \) output vector.

According to Abonyi et al. (2003) fuzzy classification can be performed as follows:

**Rule\(i\):** If \( x_1 \) is \( A_{i,1} \), \ldots, and \( x_n \) is \( A_{i,n} \) then \( g_i \), \( i = 1, \ldots, R \),

\[
\beta_i(\mathbf{x}_k) = \prod_{j=1}^{n} A_{i,j}(x_{k,j}).
\]

The rule with the highest degree of fulfillment \( \beta^* \) is declared as winner rule (i.e. winner takes all strategy). The output of the classifier is the rule consequent associated to that rule. The confidence of decision is computed as

\[
\text{cf} = \frac{\beta^*}{\sum_{i=1}^{R} \beta_i},
\]

and it can be used to fine-tune the detector. For example, a certain confidence of decision can be required to raise an alarm in order to reduce false alarms.

### 3.2. Multiobjective problems

Let us assume a MOP with \( h \) objectives \( f_i, i = 1, \ldots, h \). Let \( \mathbf{s} \) be the decision vector and \( \mathbf{S} \) the feasible region of the decision vector. That MOP can be formulated as

\[
\begin{align*}
\text{Minimize} & \quad f_1(\mathbf{s}), f_2(\mathbf{s}), \ldots, f_h(\mathbf{s}) \\
\text{s.t.} & \quad \mathbf{s} \in \mathbf{S}.
\end{align*}
\]

It is often impossible to find a solution which simultaneously minimizes all \( h \) objectives. Hence, a set of widely spread tradeoff solutions is often sought. A particular interest is on the non-dominated (Pareto optimal) decision vectors. A decision vector \( \mathbf{s}_1 \in \mathbf{S} \) is dominated by \( \mathbf{s}_2 \in \mathbf{S} \), if following formula holds:

\[
\forall i, f_i(\mathbf{s}_2) \leq f_i(\mathbf{s}_1) \quad \text{and} \quad \exists j, f_j(\mathbf{s}_2) < f_j(\mathbf{s}_1).
\]

The Pareto-optimal set is formed of non-dominated solutions and their image under the objective functions is the Pareto front (Coello, 2006; Ishibuchi and Nojima, 2007).

### 3.3. Initialization of FCs

First, C4.5 algorithm is applied to create a DT, which is then converted into an FC like presented in Abonyi et al. (2003). That can be done without decomposition error, if trapezoidal membership functions (MFs) are applied. However, applying generalized bell (gbell) MFs is beneficial, because they may have better fit to the data (Setnes and Roubos, 2000) and they have three parameters in contrast to four parameters of trapezoidal MFs. Furthermore, since their parameters can be optimized independently, standard mutation and crossover operators of MOEAs can be used without any feasibility check of MFs parameters. Thus, they are applied in this paper and defined as

\[
\mu(x; a, b, c) = \frac{1}{1 + \left|\frac{x - c}{a}\right|^{2b}}.
\]
where $x$ is the data point, and $a$, $b$, and $c$ are the parameters of an gbell MF. The value of $a$ defines the width of a MF. In this paper it is required that $a$ should be at least 1% of variable range $\gamma = ubound - lbound$, where $ubound$ and $lbound$ are, respectively, the upper and lower bounds of a variable. In addition, an MF should not be wider than $\gamma$ and therefore $0.005\gamma < a < \gamma/2$. The value of $b$ defines the fuzziness of an MF. If it is set to a high value, an MF is almost a crisp function. Moreover, if $b \approx 0$ a MF can cover large areas of universe of discourse, therefore leading to covering of fuzzy sets, which will be illustrated later in Section 3.6. Therefore, $1 < b < 10$. Center of an MF should be inside the variable range. Thus, $lbound < c < ubound$.

### 3.3.1. Simplification of initial FC

The initial FC is commonly overly complex due to the axis parallel partition of crisp DT (Abonyi et al., 2003) and the noise in real-world data. To reduce computational costs and to improve the search capabilities of MOEAs, highly similar fuzzy sets of the initial FC are merged according to the following equation (Setnes et al., 1998; Wang et al., 2005):

$$S(A_i, A_j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|} \approx \frac{\sum_{k=1}^{p} [\mu_k(x_k) \land \mu_k(x_k)]}{\sum_{k=1}^{p} [\mu_k(x_k) \lor \mu_k(x_k)]},$$

(7)

where $\cap$ and $\cup$ are the set theoretic intersection and union, respectively. Minimum is marked by $\land$ and maximum by $\lor$. The left-hand side of the formula is commonly approximated by calculating $p$ membership values $\mu$ for fuzzy sets $i$ and $j$ in discrete universe $U = \{x_k | k = 1, 2, \ldots, p\}$ (Wang et al., 2005). All pairs exceeding the threshold $A$ are merged.

Parameters of fuzzy set $A'$, which replaces fuzzy sets $A_i$ and $A_j$ are

$$d' = \max(c_i + a_i, c_j + a_j) - \min(c_i - a_i, c_j - a_j),$$

$$b' = \frac{b_i + b_j}{2} \quad \text{and} \quad c' = \frac{c_i + c_j}{2}.$$

(8)

(9)

By result of merging, the rule base may have similar (Setnes et al., 1998) or inconsistent rules and rule conditions. It is beneficial to remove them before creating the rest of the chromosomes sharing the same structure with simplified initial FC. Heuristic rule removal is applied for that purpose and will be discussed later in Section 3.4. Merging of fuzzy sets is only applied to the initial FC created by DT.

### 3.3.2. Structure of a chromosome

The simplified initial FC is the first member (chromosome) of the initial population. Its structure is coded, such that, it can be optimized using standard MOEA, such as NSGA-II. The rest $N - 1$ chromosomes, where $N$ is the population size, share the same structure with the simplified initial FC.

The structure of an FC includes two real-coded vectors defining the antecedents of the rules $A$ and the parameters of the fuzzy sets $P$. $A$ is given as

$$A = \left( A_{1,1}, A_{1,2}, \ldots, A_{1,n}, A_{2,1}, A_{2,2}, \ldots, A_{2,n}, \ldots, A_{R,1}, A_{R,2}, \ldots, A_{R,n} \right).$$

(10)

where $R$ denotes the number of rules in simplified initial FC (maximum number of rules) and $n_s$ stands for the number of variables selected from $n$ variables in initialization and simplification phases (maximum number of input variables). Naturally $n_s \leq n$, but usually $n_s < n$. $A_{ij} = \{A^*_{ij}\}$, which means that the real-coded values $A^*_{ij}$ are rounded to the nearest integer when fitness evaluation is performed (Pulkkinen and Koivisto, in press). $A_{ij} = \{0, 1, \ldots, M_j\}$ indicates which MF is used for variable $j$ in rule $i$ and $M_j$ is the number of MFs assigned to variable $j$ in simplified initial FC (maximum number of MFs in a variable). If variable $j$ is not used in rule $i$, then $A_{ij} = 0$. If rule $i$ is not used in an FC, then $\forall j, A_{ij} = 0$. However, in simplified initial FC there is no rule $i$, for which $\forall j, A_{ij} = 0$.

Parameter vector $P$ is presented as

$$P = (P_{1,1}, P_{1,2}, \ldots, P_{1,\beta}, P_{2,1}, P_{2,2}, \ldots, P_{2,\beta}, \ldots, P_{n_s,1}, P_{n_s,2}, \ldots, P_{n_s,\beta}),$$

(11)

where $\gamma$ is the number of parameters used to define a MF and $\beta = \sum_{j=1}^{n_s} M_j$ is the total number of MFs in simplified initial FC. In this paper Gbell MFs are used, so $\gamma = 3$.

Consequent part of the fuzzy rule $g = [g_1, \ldots, g_R]$ is not included into an individual. It is static and created in initialization phase by DT and by simplification operators. So, MOEA is used to select rules, rule antecedents and parameters of MFs for the pre-specified class labels. The total number of parameters $\theta$ to be optimized by MOEA is therefore given as

$$\theta = R \times n_s + \gamma \times \beta.$$

(12)

Each parameter is restricted with lower and upper bounds defined in Sections 3.2 and 3.3.2. Hence, the number of constrains is $2 \times \theta$.

### 3.3.3. Initialization of the rest of the population

The rest $N - 1$ individuals of the population are created by randomly replacing some parameters of the simplified initial FC. The replacement algorithm creates a set of widely distributed chromosomes as follows:

**Repeat for** $I = 1, \ldots, N - 1$, where $I$ is the chromosome iterator.

**Step 1**: Calculate the number of parameters to be replaced $m$ as follows:

$$m = \text{round} \left( \frac{I}{(N - 1) \times \theta} \right),$$

(13)

where round stands for the operator rounding the result to the nearest integer.

**Step 2**: Choose randomly $m$ parameters out of $\theta$. 

Step 3: Replace them by randomly generating \( m \)
parameters between their corresponding limits, defined in Sections 3.2 and 3.3.2.

End for

So, a population of widely distributed chromosomes is created. They all share the same structure with the simplified initial FC, which was defined in Section 3.3.2. The rule base of the rest \( N - 1 \) chromosomes may contain inconsistencies due to the random replacement algorithm, so they go through heuristic rule reduction presented next.

### 3.4. Heuristic rule and rule condition reduction

It is beneficial to remove rules and rule conditions heuristically in order to guide and speed up the evolutionary search. The following three heuristics are applied in this paper:

1. If there are rules with exactly the same antecedent part, all but one of them are removed (Setnes et al., 1998). The preserved rule is randomly selected.
2. There can be rules of different length in which all conditions of the shorter rule(s) are present in the longer rule(s). Those rules are inconsistent. The longer rule(s) will never obtain higher degree of firing than the shorter rule(s), because the T-norm in this paper is product. Out of those inconsistent rules only one rule is preserved. By uniform chance, the preserved rule is either the longest rule (i.e. the most specific rule) or it is randomly selected out of the inconsistent rules.
3. If there are conditions, which are present in all of the rules, they are removed from all of them (Pulkkinen and Koivisto, 2007).

These heuristics are applied to whole initial population. Furthermore, during MOEA optimization they are applied to whole offspring population. Thus, there are no inconsistencies in the rule base in any of the chromosomes.

When the initial FC generated by DT is simplified by merging of fuzzy sets and by removing rules and rule conditions, the number of parameters \( \theta \) to be optimized is decreased. But because the whole population shares the same structure defined in Section 3.3.2, \( \theta \) is not affected when the other FCs (i.e. the \( N - 1 \) FCs of initial population and the whole offspring population) go through heuristic rule and rule condition reduction. Then, simply the removed conditions are set to zero.

### 3.5. Metric for classifier's performance

It is a known fact that accuracy (i.e. the proportion of correctly classified data points to the total number of data points) is not a good metric for classifier’s performance when misclassification costs and/or class distributions are not known (Provost et al., 1998; Fawcett, 2001; Setzkorn and Paton, 2005; Ben-David, 2007). In this subsection, a simple example is given to illustrate that. Furthermore, the performance metric used in this paper is presented.

Consider the well-known Wisconsin breast cancer data (WBCD) from UCI repository (Newman et al., 1998). It contains 458 data points, which should be classified as benign (normal) and 241 data points, which should be classified as malignant (alarm). Let us assume that two classifiers are created for that problem; classifier A with accuracy 65.52% and classifier B with accuracy 49.36%. One may instantly conclude that classifier A is more accurate. However, when their confusion matrices (see Tables 1 and 2) are considered, decision about their superiority is not so straight forward.

Classifier A is a major class classifier, so it always outputs the major class (benign), and never the minor class (malignant). Therefore, it does not distinguish between benign and malignant cancers. Classifier B is a random classifier; therefore, it equals flipping the coin for two class classification problem, like WBCD. The superiority of the classifiers depends on the misclassification costs. However, based on this simple example, it is obvious that the misclassification rate is not a good metric to use, when misclassification costs and/or class distributions are not known. One good alternative metric is Cohen’s kappa, which describes the proportion of the classifier’s merits to the total number of classifications which are not attributed by chance alone. It is commonly used in psychology, biology, statistics and medicine, however, in Machine Learning it is rarely used (Ben-David, 2007). It is a good metric, especially when the number of different classes is high and therefore receiver operating characteristics (ROC) curves (Provost and Fawcett, 1997) are hard to visualize (Ben-David, 2007). In our case, however, when the number of classes is two, computing the TP rate, also called hit rate, and FP rate, also called false alarm rate, are used because of their intuitiveness.

According to (Fawcett, 2001) TP rate and FP rate are given as follows:

\[
\text{TP rate} = \frac{\text{positives correctly classified}}{\text{total positives}},
\]

Table 1

<table>
<thead>
<tr>
<th></th>
<th>Predicted as benign</th>
<th>Predicted as malignant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign</td>
<td>458</td>
<td>0</td>
</tr>
<tr>
<td>Malignant</td>
<td>241</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2

<table>
<thead>
<tr>
<th></th>
<th>Predicted as benign</th>
<th>Predicted as malignant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benign</td>
<td>222</td>
<td>236</td>
</tr>
<tr>
<td>Malignant</td>
<td>118</td>
<td>123</td>
</tr>
</tbody>
</table>
MOEAs are applied, a set of Pareto-optimal solutions presenting tradeoff between TP and FP rates is obtained, from which user can select a solution based on his/her preferences.

3.6. Interpretability of FCs

Often interpretability of FCs is measured by calculating the number of rules and total number of conditions in rules (total rule length) (Ishibuchi et al., 2001; Setzkorn and Paton, 2005; Ishibuchi and Nojima, 2007). Those metrics, however, do not indicate whether fuzzy partition is transparent or not. One may question, that whether adding the number of fuzzy sets as an objective would solve that problem. Naturally, when the number of fuzzy sets per variable is minimized, it is more likely that fuzzy partition is transparent. However, there can be as little as two fuzzy sets assigned to a certain variable, but if a fuzzy set is covered by another fuzzy set, the partition is not transparent (Wang et al., 2005). Moreover, there may be gaps between fuzzy sets, which deteriorates the linguistic interpretability.

Instead of those aforementioned metrics, slightly modified versions of interpretability metrics proposed in Kim et al. (2006), namely the length of overlap and the length of discontinuity between fuzzy sets, are used. In a nutshell, it is desired that the intersection value of two fuzzy sets would lie between user specified constants $z_L$ and $z_H$. If the intersection value is higher than $z_H$, overlap penalty $P_{OL}$ is added, whereas if it is less than $z_L$, discontinuity penalty $P_{DC}$ is added.

However, those penalties are not a guarantee for transparent fuzzy partition. In cases of complete or restricted covering (Wang et al., 2005) (e.g. a wide fuzzy set covers a narrow fuzzy set), a minor penalty is given despite the fact that the fuzzy partition is far from transparent. Furthermore, when relaxed covering (Wang et al., 2005) occurs, it is possible that no penalty at all is given (see Fig. 5c). Also, the commonly used similarity measure for fuzzy sets in formula (7) is not informative in cases of covering (Wang et al., 2005; Paiva and Dourado, 2004). Therefore, the middle value penalty $P_{MV}$ is introduced to deal with the issues of covering.

The rest of this subsection is organized as follows. First, calculating $P_{OL}$ and $P_{DC}$ is briefly presented, and reader is referred to Kim et al. (2006) for more information. Finally, computing $P_{MV}$ is introduced.

### 3.6.1. Overlap penalty

To calculate the length of overlap for fuzzy sets, left and right intersection points for the user specified level $z_H = 0.6$ (the same value as in Kim et al., 2006) are computed first for all fuzzy sets. Since $gbell$ MFs are applied, the intersection points can be computed as

$$I_L(z) = c - a \left(1 - \frac{z}{a}\right)^{1/2b}$$  \hspace{1cm} (16)
and
\[ I_R(x) = c + a \left( \frac{1 - x}{2} \right)^{1/2} b \].

(17)

\( P_{OL} \) is computed according to Kim et al. (2006):
\[ P_{OL} = \frac{1}{n_s} \sum_{i=1}^{n_s} \frac{N_{ov}^i}{\sum_{j=1}^{l_i} \lambda_{ij}}, \]

(18)
where \( \lambda_{ij} \) is the length of \( j \)th overlap between two MFs in input variable \( i \). It is computed using the left and right intersection points, like illustrated in Fig. 5(a). \( N_{ov}^i \) is the number of MF pairs in input variable \( i \), which may overlap. More formally \( N_{ov}^i \) is defined as
\[ N_{ov}^i = \left( \frac{M_i}{2} \right) = \frac{M_i!}{2(M_i - 2)!}, \]

(19)
where \( M_i > 2 \) stands for the number of active fuzzy sets in input variable \( i \). If there are only 2 MFs, \( N_{ov}^i = 1 \). \( P_{OL} \) is not calculated for a certain variable, if the number of active MFs assigned to it is less than 2.

3.6.2. Discontinuity penalty

Similarly to \( P_{OL} \), computing \( P_{DC} \) is started with computing the left and right intersection points for the user specified level \( a = 0 \) (the same value as in Kim et al., 2006) using formulae (16) and (17). Then, \( P_{DC} \) is computed slightly differently than in Kim et al. (2006) as proportion of total length of discontinuity to range \( \zeta \):
\[ P_{DC} = \frac{1}{n_k} \sum_{i=1}^{n_k} \sum_{j=1}^{G_i} \frac{\psi_{ij}}{l_i}, \]

(20)
where \( G_i \) is the number of discontinuities and \( \psi_{ij} \) is the length of the \( j \)th discontinuity present in input variable \( i \), respectively (see also Fig. 5(b)). \( P_{DC} \) is not calculated for a certain variable, if the number of active MFs assigned to it is 0.

3.6.3. Middle value penalty

\( P_{MV} \) is added to prevent relaxed covering of MFs. In Fig. 5(c) an example of relaxed covering is shown and based on \( P_{OL} \) and \( P_{DC} \) no penalty is given, even the partition is not transparent. In this paper the effects of this problem are reduced by adding \( P_{MV} \) if the value of another fuzzy set is higher than \( a \) in the center of another fuzzy set (see Fig. 5(c)):
\[ P_{MV} = \frac{1}{n_k} \sum_{i=1}^{n_k} \delta_i, \]

(21)
where
\[ \delta_i = \begin{cases} \delta_i^* - a_L & \text{if } \delta_i^* > a_L, \\ 1 - a_L & \text{if } \delta_i^* \leq a_L, \end{cases} \]

(22)
where \( \delta_i^* \) is the maximum middle value present in variable \( i \):
\[ \delta_i^* = \max_{j \neq k} \left( \mu_{j,k}(c_k; a_j, b_j, c_j) \right), \]

(23)
where \( G_i \) is the number of discontinuities and \( \psi_{ij} \) is the length of the \( j \)th discontinuity present in input variable \( i \), respectively (see also Fig. 5(b)). \( P_{DC} \) is not calculated for a certain variable, if the number of active MFs assigned to it is less than 2.
3.7. Overall fitness function

Naturally, TP rate, FP rate, number of rules, number of conditions, $P_{OL}$, $P_{DC}$, and $P_{MV}$ could be separate objectives to be minimized by MOEAs. However, when the number of objectives increases, the number of non-dominated chromosomes in population increases as well. That decreases the selection pressure and deteriorates the search efficiency of all MOEAs using Pareto-ranking as selection method (e.g. NSGA-II) (Purshouse and Fleming, 2003; Hughes, 2005). Two or three objectives are commonly optimized simultaneously; however, in some applications up to nine objectives are used. Only in few applications more than nine objectives are considered (Deb and Saxena, 2006).

In this paper, therefore, the objectives to be minimized need to be selected carefully. Since TP and FP rates are the crucial requirements for the detector to meet, they need to be selected as objectives. Transparency of fuzzy partition is another important objective. Thus, transparency penalty $T$:

$$T = P_{OL} + P_{DC} + P_{MV}$$

(24)

needs to be minimized. It will be shown later in Section 4 that by minimizing $T$, usually the number of rules and rule conditions are also reduced. Thus, they are not selected as objectives like in many other studies (e.g. Ishibuchi et al., 2001; Setzkont and Paton, 2005; Ishibuchi and Nojima, 2007). Hence, solutions with extremely small rule base, say with one rule and one rule condition, do not survive only because of their small rule base. That increases the selection pressure and improves the quality of solutions.

So, the three objectives to be minimized are

$$o_1 = 1 - TP \text{ rate, } o_2 = FP \text{ rate, } o_3 = T.$$  

(25)

To avoid impractical solutions, FP and TP rates are constrained inside a square, whose side is $d$. In this paper it is required that TP rate is at least 0.5 and FP rate the most 0.5. Thus, $d = 0.5$. The normalized constraints are

$$\text{constraint}_1 = \frac{d - o_1}{1 - d} \geq 0, \quad \text{constraint}_2 = \frac{d - o_2}{1 - d} \geq 0.$$  

(26)

4. Results

The collected and preprocessed data contained 10268 data points. Each data point consisted of four input variables and a class label (alarm or normal). Variables are labeled as A, B, C and D for confidentiality reasons. Data were divided into train and test sets. FCs were identified using only the train data, which contained 80% of data, that is, 8214 data points. FCs were tested with the rest 2054 unseen data points. The proposed hybrid GFS was run with the parameters shown in Table 4, and a vast amount of Pareto-optimal FCs was obtained.

![Fig. 6 presents the initial fuzzy sets obtained by transforming DT into an FC. It is seen that the fuzzy partition is far from transparent due to many highly overlapping fuzzy sets and from Table 5 it is noticed that the transparency penalty $T$ is 1.062. Similar fuzzy sets were merged with merging threshold $\Delta = 0.25$, which was the same value as in Setnes et al. (1998). That resulted into more transparent partition (see Figs. 7 and 8) and decreased $T$ into 0.755. Also, the number of rules was reduced from 47 to 21. Those simplification operations decreased TP rate from 0.888 to 0.049, but also decreased FP rate from 0.200 to 0.020. This kind of TP rate is obviously not acceptable and also $T$ should be decreased. That was obtained by MOEA optimization. Because the search space was highly reduced due to appropriate initialization, very good results were obtained, which is visualized by sketching the obtained Pareto fronts for the train and test sets (see Figs. 9 and 10). In those Figs. only the solutions having at least 0.7 FP rate and at most 0.3 FP rate are presented. From the obtained solutions, a solution based on the current preferences can be selected, and the rest of the solutions may be stored for the possible usage. The solutions were widely spread; there were FCs with TP rate 1, FCs with FP rate 0.009, and also highly transparent FCs with transparency penalty 0. However, due to the contradicting nature of the objectives, all of those extreme values were not present in a single FC.

During the MOEA optimization the average number of rules was reduced from 15.641 to 5.599 and the average number of rule conditions from 54.286 to 10.495. That happened even though those were not used as fitness objectives, which means that they are somewhat correlated with the average value of $T$ which was reduced from 1.095 to 0.099 during the MOEA optimization.

An FC was selected for further analysis from the set of Pareto-optimal solutions based on the current needs. Its fuzzy partition is presented in Fig. 8 and its TP = 0.906, FP = 0.048 and $T = 0.003$. It consisted of only seven rules and 13 rule conditions. In Section 4.2 its performance is analyzed more thoroughly.

4.1. Results comparison

To study whether the obtained good results were due to efficient MOEA algorithm, NSGA-II, or combination of

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Parameters for experimental setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>1000</td>
</tr>
<tr>
<td>Number of generations</td>
<td>1000</td>
</tr>
<tr>
<td>Distribution index for mutation</td>
<td>20</td>
</tr>
<tr>
<td>Distribution index for crossover</td>
<td>20</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>0.9</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>$1/\theta$</td>
</tr>
<tr>
<td>Pruning confidence (C4.5)</td>
<td>5</td>
</tr>
</tbody>
</table>

Exactly the same parameters were used in our earlier study (Pulkkinen and Knivisto, in press). The same crossover and mutation probabilities and distribution indexes were also applied in Deb et al. (2002).
heuristic rule and rule condition removal, simplification operators and NSGA-II algorithm, we identified FCs with and without those operators. The GFS which was applied in previous subsection using both heuristic rule and rule conditions removal and simplification operators is denoted here as GFS1. GFS2 uses heuristic rule and rule conditions removal but does not perform simplification operators, whereas GFS3 performs simplification operators but does not apply heuristic rule and rule conditions removal. Finally, GFS4 applies neither of those operators.

The experimental setup was kept exactly the same as in previous subsection. Furthermore, for all GFS versions the initial FC and random seed were the same in order to examine the effect of the aforementioned operators. The experiments were performed on a desktop Pentium 4 3 GHz PC with 1 GB of RAM, and the execution times of each GFS version were recorded.

Fig. 11 presents TP and FP rates for test data set for the FCs obtained by different GFS versions. In this figure only the high quality solutions are presented. It is required that TP rate is at least 0.9 and FP rate at most 0.1. Moreover, transparency penalty $T$ is required to be less than 0.2. It is noticed that there are no FCs identified by GFS4 meeting these requirements. Moreover, the most accurate results are obtained by GFS1.

In Table 6 comparison of execution times of different GFS are shown. It is seen that GFS1 has clearly the shortest execution time, whereas GFS4 has clearly the longest execution time. There is no significant difference in the execution times of GFS2 and GFS3. The differences in execution times are due to reduced computational costs of fitness evaluations by heuristic rule and rule conditions removal and simplification operators. Applying these operators make FCs simpler and therefore reduces the computational costs of fitness evaluations. However, these operators also improve the quality of FCs as was noticed in Fig. 11.

4.2. Insights to the bioaerosol detector and guidelines for further development

Another contribution of this work is that it provided the guidelines for further development of the bioaerosol detector. Fig. 12 presents the performance of the selected FC for test data. It is seen that all harmful particles were detected; however, some false alarms were also raised. Next, the reasons for false alarms are discussed.
A very intuitive and effective way to analyze the behavior of the selected FC is to sketch its output along with the measurement values. By analyzing those figures, keeping in mind the function of the interpretable fuzzy rules, it was clearly indicated that some particles were easily recognized as normal or alarm classes. However, there were
some harmless particles, which clearly met the conditions of an alarm. When the problem was further analyzed, it was shown that actually the selected FC was able to classify the particles correctly in the beginning of the spreading period. However, in the middle of the spreading period the other three variables returned to their zero values, but variable A remained high. That caused the false alarms. An example of that problem is illustrated in Figs. 13 and 14.

The reason for the false alarms was very clear, thanks to the transparent rule base and the explorative data analysis. Therefore, straightforward guidelines for further development of bioaerosol detector were given and applied.
4.3. Interpretable model makes fine-tuning easier

Like mentioned earlier, the bioaerosol detector is in prototype phase and changes are made frequently. After the first measurement campaign, significant improvements to the detector were made, which made it more reliable. Due to the improvements, however, the absolute values of measurements were significantly changed, while the measuring mechanism and logic remained approximately the same. Besides that, a very limited amount of data for the improved bioaerosol detector was available, because no measurement campaigns with it was yet performed.

It was, however, important to prove the potential of the bioaerosol detector in practice in order to continue the funding of the project. Unfortunately, the earlier developed FCs were not able to differentiate between harmful and safe particles due to the changes in the absolute values of the data. Naturally, if it had been known how much the values have changed, the previously collected data could have been modified and the supervised algorithm run again. Changes in values, however, were not specifically known and there was no time for deep analysis because of the strict time limitations. Therefore, the highly interpretable fuzzy sets were manually tuned\(^4\) using the available limited data set and that took only few hours. If black-box models had been used instead, fine-tuning without a new measurement campaign would not have been possible.

5. Conclusions

The aim of this work was to develop a model, which can be used as a reasoning mechanism in a bioaerosol detector. The requirements for the model were high true positive (TP) rate and low false positive (FP) rate. In addition, it was important for the sake of confidence in the model and of further development of the detector, that the developed model is as interpretable as possible. Therefore, the problem at hand was a MOP with conflicting objectives.

As an identification framework, a hybrid genetic fuzzy system (GFS) was applied. The population was initialized with the help of a crisp decision tree (DT) algorithm. That was clearly a better starting point for further optimization than commonly used random initialization. However, the initial fuzzy classifier (FC) was overly complex due to crispness of DT and the noise in the real-world data. Thus, merging of similar fuzzy sets was applied, which led to some similar and inconsistent rules. Hence, heuristic rule and rule conditions removal took place. After that, the rest of population was created, such that the population was

\(^4\)Since the measuring mechanism and logic had remained approximately the same, it was sufficient to tune only the fuzzy sets without modifying the rule base.
highly distributed and all individuals of population shared the same structure with the simplified initial FC.

Heuristic rule and rule conditions removal was performed during the MOEA optimization, which both improved the quality of solutions and reduced the execution time. The fitness of chromosomes was evaluated based on their FP and TP rates and transparency of their fuzzy partition. Even though the number of rules and rule conditions were not used as fitness objectives, the final population contained significantly less rules and rule conditions than the initial population. Furthermore, a vast amount of FCs with good tradeoff between the objectives was found. Because transparency of fuzzy partition was used as an objective, the obtained fuzzy partitions and rules were highly interpretable.

Another contribution of this work was that it brought insights for the further development of bioaerosol detector. By analyzing the false alarms, keeping in mind the interpretable fuzzy rules, it was found out that some of them were caused by dysfunction of bioaerosol detector. Straightforward guidelines for further development were therefore given.

This is an ongoing project; therefore, it was important to prove the potential of proposed bioaerosol detector in

---

Fig. 12. Performance of the selected FC for test data. Classes 1 and 2 indicate normal and alarm, respectively. It is seen that all harmful particles were detected, however, there were also some false alarms.

Fig. 13. First, particles of fog oil and smoke Grenade are in the air (samples 2160–2200), which causes a rise in variables A and B. However, after that variable B returns to its zero value, but variable A still stays in higher value. That causes the false alarm. Later (samples 2360–2385), some harmful particles are spread and an alarm is correctly raised.
order to continue the funding of the project. The results of
this paper confirmed its capability in differentiating
between some safe and harmful aerosols. Therefore its
potential was proven.

Finally, changes in bioaerosol detector are frequent due
to its constant development. After the first measurement
campaign, the fuzzy partition needed to be modified to
correspond the behavior of the further developed bioaero-
sol detector. That was, however, possible in short time,
thanks to the transparency of the FCs.

References

compact, accurate, and linguistically-sound fuzzy classifiers based on a
decision-tree initialization. International Journal of Approximate
Reasoning 32 (1), 1–21.

Ben-David, A., 2007. A lot of randomness is hiding in accuracy.
Engineering Applications of Artificial Intelligence 20 (7), 875–885.

Coello, C.C., 2006. 20 years of evolutionary multiobjective optimization:
what has been done and what remains to be done. In: Yen, G.Y.,
Fogel, D.B. (Eds.), Computational Intelligence: Principles and
Practice. IEEE Computational Intelligence Society, pp. 73–88.

Ten years of genetic fuzzy systems: current framework and new trends.

through dimensionality reduction for certain large-dimensional multi-
objective optimization problems. In: IEEE Congress on Evolutionary
Computation, Vancouver, BC, Canada, pp. 3353–3360.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist
multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on
Evolutionary Computation 6 (2), 182–197.

Elder, J.F., Pregibon, D., 1996. A statistical perspective on knowledge
discovery in databases. In: Fayyad, U., Piatetsky-Shapiro, G., Smyth,
P., Uthurusamy, R. (Eds.), Advances in Knowledge Discovery and
Data Mining, pp. 83–113.

International Conference on Data Mining, pp. 131–138.

fuzzy modeling. In: 6th European Congress on Intelligent Techniques

real-time detection of bioaerosols using simultaneous measurement of
particle aerodynamic size and intrinsic fluorescence. Journal of
Aerosol Science 28 (3), 471–482.

Haubelt, C., Gamenik, J., Teich, J., 2005. Initial population construction
for convergence improvement of MOEAS. In: Coello, C.A.C.,
Aguirre, A.H., Zitzler, E. (Eds.), EMO 2005, Lecture Notes in

Hughes, E.J., 2005. Evolutionary many-objective optimisation: many once
or one many? In: The 2005 IEEE Congress on Evolutionary

tradeoff of fuzzy systems by multiobjective fuzzy genetics-based
machine learning. International Journal of Approximate Reasoning
44 (1), 4–31.

Ishibuchi, H., Nakashima, T., Murata, T., 2001. Three-objective genetics-
based machine learning for linguistic rule extraction. Information
Sciences 136 (1–4), 109–133.

Ishibuchi, H., Nojima, Y., Kuwajima, I., 2006. Fuzzy data mining by
heuristic rule extraction and multiobjective genetic rule selection. In:
2006 IEEE International Conference on Fuzzy Systems, Vancouver,
BC, Canada, pp. 7824–7831.

Janka, K., Reinivaara, R., Enroth, J., Hytönen, J., Tikkanen, J., Rostedt,
A., Putkiranta, M., Marjamäki, M., Laaksonen, J., Keskinen, J.,
Humpi, T., 2007. UV-fluorescence-detection based bioaerosol-warn-
ing concept with a background-aerosol sensor. In: The 9th Symposium
on Protection against Chemical and Biological Warfare Agents,
Gothenburg, Sweden.

interpretable Takagi-Sugeno fuzzy models with a new encoding
scheme. IEEE Transactions on Systems, Man, and Cybernetics—
Part B: Cybernetics 36 (5), 1006–1023.

computing for intelligent data analysis. In: 18th International
Conference of the North American Fuzzy Information Processing
Society, pp. 527–531.

Newman, D., Hettich, S., Blake, C., Merz, C., 1998. UCI repository of
machine learning databases. URL [http://www.ics.uci.edu/~mlearn/
MLRepository.html).

Paiva, R.P., Dourado, A., 2004. Interpretable and learning in neuro-

Poles, S., Fu, Y., Rigoni, E., 2006. The effect of initial population sampling
on the convergence of multi-objective genetic algorithms. In:
MOPGP’06: 7th International Conference on Multi-Objective Program-
mig and Goal Programming, Loire Valley (City of Tours), France.

performance: comparison under imprecise class and cost distributions.
In: 3rd International Conference on Knowledge Discovery and Data
Mining, KDD-97, pp. 43–48.


A Genetic Fuzzy System with Inconsistent Rule Removal and Decision Tree Initialization

Pietari Pulkkinen and Hannu Koivisto

Abstract. This paper presents a genetic fuzzy system for identification of Pareto-optimal Mamdani fuzzy models (FMs) for function estimation problems. The method simultaneously optimizes the parameters of fuzzy sets and selects rules and rule conditions. Selection of rules and rule conditions does not rely only on genetic operators, but it is aided by heuristic rule and rule conditions removal. Instead of initializing the population by commonly used Wang-Mendel algorithm, we propose a modification to decision tree initialization. Experimental results reveal that our FMs are more accurate and consist of less rules and rule conditions than the FMs obtained by two recently published genetic fuzzy systems [2, 3].

1 Introduction

One reason for the popularity of fuzzy models (FMs) is that they can be transparent if identified adequately. Recent trend has been the application of multiobjective evolutionary algorithms (MOEAs) to find FMs presenting trade-off between accuracy and interpretability criteria (see for example [2, 3, 11, 14, 16]). Those methods are also called genetic fuzzy systems (GFS) [4]. It was stated in [2] that most of MOEA based methods in literature consider only classification problems, that is, the output belongs to a set of pre-specified labels. Only few have covered function estimation problems, that is, the value of output is continuous (see for example
Several studies ([14], [16]) have applied Takagi-Sugeno (TS) FMs, which are considered harder to interpret than Mamdani FMs due to their linear consequent part. In [2, 3], MOEAs were applied to find a Pareto-optimal set of Mamdani FMs. In [2], the number of rules and parameters of fuzzy sets were optimized by MOEA, but the rule conditions were kept fixed. Thus, optimal rules can not be found. In [3], the number of rules and rule conditions were optimized by MOEA, however, the parameters of fuzzy sets were fixed and pre-specified. Therefore, fuzzy partitions may not present the real distribution of data and the accuracy of FMs is deteriorated ([7]). On the other hand, it was pointed out in [8] that if the fuzzy partitions of each variable are available by a priori knowledge, modification of them can impair the interpretability of FMs.

When function estimation problems are considered, the initial population is usually obtained randomly or by Wang and Mendel (WM) method ([15]). Naturally random initialization does not guarantee a good starting point for further optimization. WM method also has several drawbacks. First, it leads to a high number of rules when high-dimensional problems are considered. Second, it requires that each variable is partitioned with fuzzy sets a priori. Third, it uses all available variables in all rules, thus leading to unnecessary complexity of the rule base.

DT based initialization algorithm was introduced in [1]. Because it can automatically select the relevant variables and partition the input space, it makes it a desirable initialization algorithm. In [1] it was applied to classification problems and here we modify it to suit for function estimation problems.

The further optimization is performed by MOEA, which selects the adequate rules and rule conditions and optimizes the parameters of fuzzy sets. Rule selection is not solely based on genetic operators but aided by heuristic removal of inconsistent rules and rule conditions. Our method is validated by identifying FMs for four well-known problems and comparing our results to other MOEA based approaches ([2, 3]). Our results show that our method obtains more compact and accurate FMs than in the comparative studies.

This paper is organized as follows. First, Mamdani FMs are defined. Then, in section 3 our identification method is introduced. After that, in section 4 the results comparison is performed. Finally, conclusions are given in section 5.

2 Mamdani Fuzzy Models

Let the dataset with $D$ data points and $n$ input variables be denoted as $Z = [X, y]$, where $X$ is $D \times n$ input matrix and $y$ is $D \times 1$ output vector. Mamdani fuzzy rules are denoted as:

$$R_i : \text{If } x_1 \text{ is } A_{i,1}, \ldots, x_n \text{ is } A_{i,n} \text{ then } B_i,$$

where $A_{i,j}, \ j = 1, \ldots, n, \ i = 1, \ldots, R$, is an input membership function (MF), $B_i$ is an output MF, and $R$ is the number of rules. In order to reduce the computational costs, the output of FMs is computed here by approximation of centroid method ([17], [3]):
\[ \hat{y}_k = \frac{\sum_{i=1}^{R} \beta_i(x_k) \bar{B}_i}{\sum_{i=1}^{R} \beta_i(x_k)}, \]  
(1)

where \( \bar{B}_i \) is the center an output MF \( B_i \), and \( \beta_i \) is the rule activation degree: \( \beta_i(x_k) = \prod_{j=1}^{n} A_{i,j}(x_{k,j}) \). When output MFs are uniformly and symmetrically shaped, eq. (1) is an equivalent to centroid defuzzification [17].

3 Proposed Identification Method

3.1 Mamdani FM Initialization Using C4.5 Algorithm

In [1], C4.5 algorithm [12] was used to initialize fuzzy classifiers. Since we consider function estimation problems, the output data need to be discretized in order to use C4.5 algorithm. That is done by dividing the output to \( N_{\text{out}} \) crisp regions, where \( N_{\text{out}} \) is a positive integer. Each output value falls into one of these \( N_{\text{out}} \) regions and it is replaced with corresponding class label \( S \in \{1, \ldots, N_{\text{out}}\} \), representing these regions. Then, C4.5 is used to match the input vectors to one of these class labels.

In Fig. 1 Box-Jenkins Gas Furnace data [1] is used as an example of applying C4.5 to predict continuous output. It is seen that crisp decision tree (DT) can not interpolate between rules. Thus, the resulting DT is transformed into a fuzzy classifier (FC) using the methodology given in [1]. However, the class labels in rule consequent are replaced with membership functions (MFs) in order to predict continuous outputs. To be able to cover the whole output range, the center of an output MFs need to be placed to lower \( \chi_l \) and upper \( \chi_u \) bounds of the range \( \chi = \chi_u - \chi_l \). For the sake of interpretability, the MFs centers are evenly placed between lower and upper bounds:

\[ c_1 = \chi_l \text{ and } c_j = c_{j-1} + \frac{\chi}{N_{\text{out}}-1}, \quad j = 2, \ldots, N_{\text{out}}. \]  
(2)

Center values of output MFs are one of those centers \( \bar{B}_i = c_{S_i} \), where \( S_i \in \{1, \ldots, N_{\text{out}}\} \), \( i = 1, \ldots, R \). Since \( \bar{B}_i \) is the only parameter needed in eq. (1) to compute the output of an FM, the type of an output MF can be, for example, triangular, Gaussian, singleton, or generalized bell (gbell) without affecting the result of computation. An example of an output partition with 5 Gbell MFs is presented in Fig. 1 (c). Gbell MFs are defined as:

\[ \mu(x; a, b, c) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2b}}, \]  
(3)

where \( b \) defines the fuzziness of an MF. Like mentioned earlier, it has no effect on computation of the output of an FM and in Fig. 1 (c) it was set to 2.5. For the sake of interpretability, it is required that at the intersection points, the intersecting MFs have membership value of 0.5. Therefore, the distance from the center \( c_j \) of each MF to the intersection points is \( a = \frac{\chi}{2(N_{\text{out}}-1)} \).

\[ ^1 \text{ That dataset is described later in section 4.1} \]
3.2 Merging of Fuzzy Sets

The FM initialized by C4.5 is likely to contain highly similar fuzzy sets and it is beneficial to merge them. As a similarity measure the following is used: \( S(A_i, A_j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|} \), where \( \cap \) and \( \cup \) are the set theoretic intersection and union, respectively. In this paper all pairs exceeding the user specified threshold \( \Delta \) are merged. After merging, the resulting FM may have similar or inconsistent rules in the rule base. Therefore it goes through heuristic removal of inconsistent rules and rule conditions described later in section 3.5. Then, the rest of the population is created by randomly replacing some parameters of the simplified initial FM, such that, the initial population is widely spread. Finally, the rest of the population go through the heuristic rule and rule conditions removal and the initial population is ready for further optimization by MOEA performed by popular NSGA-II.

3.3 Coding of the Mamdani FM

Mamdani FM is presented with three real-coded vectors: antecedent vector \( A^* \), input MF parameter vector \( P \), and output MF parameter vector \( S^* \). Asterisks in \( A^* \) and \( S^* \) indicate that although real-coded values are optimized by MOEA, the values are rounded to the nearest integer when fitness evaluation is performed. The resulting integer vectors are respectively marked as \( A \) and \( S \).

\( A \) specifies for each rule \( i = 1, \ldots, R \), which MF is used for variable \( j = 1, \ldots, n_s \), where \( n_s \) is the number of variables used in simplified initial FM. Because C4.5 can select input variables, \( n_s \leq n \):

\[
A = (A_{1,1}, A_{1,2}, \ldots, A_{1,n_s}, \ldots, A_{R,1}, A_{R,2}, \ldots, A_{R,n_s}),
\]

(4)

\( A_{i,j} \in \{0, 1, \ldots, M_j\} \), where \( M_j \) stands for the number of MFs in variable \( j \) in simplified initial FM. If \( A_{i,j} = 0 \), variable \( j \) is not used in rule \( i \). Variable selection
and rule selection are possible during the evolutionary search. Variable \( j \) is removed if \( \forall i, A_{i,j} = 0 \) and rule \( i \) is removed if \( \forall j, A_{i,j} = 0 \).

\( P \) defines the parameters of input MFs:

\[
P = (P_{1,1}, P_{1,2}, \ldots, P_{1,\beta}, \ldots, P_{\gamma,1}, P_{\gamma,2}, \ldots, P_{\gamma,\beta}),
\]

where \( \gamma \) stands for the number of parameters used to define an MF and \( \beta = \sum_{j=1}^{n_s} M_j \) is the total number of MFs in simplified initial FM. Because Gbell MFs are applied in this paper, \( \gamma = 3 \).

The Gbell parameter \( a \) is constrained between \((0.005\chi, \chi/2)\), which means that the width of an MF should be at least 1% of the variable range \( \chi \) and not wider than \( \chi \). Parameter \( b \) defining the fuzziness of an MF is required to be \( > 1 \) so that it does not cover large areas with low degree of membership, which leads to poor distinguishability of MFs [9]. On the other hand, large \( b \) value makes MFs almost crisp. Therefore \( 1 < b < 10 \). Finally, the center of an MF is required to be inside the variable range. Thus, \( \chi_l < c < \chi_u \).

During the evolutionary learning the antecedents of the rules and parameters of input MFs may be altered, such that, the initial output MFs may not be appropriate anymore [11]. Hence, \( S = (S_1, S_2, \ldots, S_R) \) stating the output MFs of the rules (see section 3.1) is optimized during the evolutionary search.

The total number of parameters to be optimized \( \theta \) by MOEA is the sum of the lengths of \( A^*, P \), and \( S^* \). Thus, \( \theta = R \times n_s + \gamma \times \beta + R \).

### 3.4 MOEA Optimization of the Initial Population

Each member of the population has the same amount of parameters to be optimized as in initial FM created by C4.5 and simplification operators. Fig. 2 depicts the further optimization of the initial population resulting to a set of Pareto-optimal FMs. Although C4.5 algorithm was used with discretized output data, during the further optimization the original undiscretized data are used. The fitness objectives to be minimized are: number of rules \( R \), number of rule conditions (total rule length) \( R_{\text{cond}} \), and mean squared error \( \text{MSE} = \frac{1}{D} \sum_{k=1}^{D} (y_k - \hat{y}_k)^2 \).

### 3.5 Heuristic Rule and Rule Condition Removal

At every iteration, NSGA-II evaluates the fitness of the offspring population [6]. The size of the offspring population is the same as the population size \( N_{\text{pop}} \). Thus, the fitness of \( N_{\text{pop}} \) FMs is evaluated, which is a computationally demanding operation. If these FMs contain inconsistent rules and rule conditions, the fitness evaluations take a longer time than in case those inconsistencies are absent. Therefore the evolutionary search is aided by three heuristic operations: (1) Only one of the rules with exactly the same antecedent part is preserved [13]. (2) If there are rules of different length in which all conditions of the shorter rule(s) are present in the longer rule(s),
only one of them is preserved. By uniform chance, the preserved rule is either the longest rule (i.e. the most specific rule) or it is randomly selected out of the inconsistent rules \[9\]. (3) If there are conditions, which are present in all of the rules, they are removed from all of them \[10\].

4 Experiments

4.1 Datasets

The aim in Electric dataset is to estimate the maintenance costs of a medium voltage electrical network in a town \[5\] based on 4 input variables. It is a real-world dataset consisting of 1059 datapoints. Data partitions for 5-fold cross-validation were obtained from the website of Jorge Casillas: http://decsai.ugr.es/~casillas/fmlib/.

Mackey-Glass (MG) is a chaotic time series defined by:

\[
\frac{dx(t)}{dt} = \frac{0.2x(t - \tau)}{1 + x^{10}(t - \tau)},
\]

The problem is to predict \(x(t + 6)\) based on \([x(t), x(t - 6), x(t - 12), \text{and } x(t - 18)]\). In \[3\] and also in our experiments, \(\tau\) was set to 17 and \(x(0)\) to 1.2, which are typical values in the literature. Like in \[3\], we generated 500 datapoints to be used in our experiments.

Lorenz attractor (Lorenz) is another chaotic time series:

\[
\frac{dx(t)}{dt} = \sigma(y(t) - x(t)), \quad \frac{dy(t)}{dt} = \rho x(t) - y(t) - x(t)z(t), \quad \frac{dz(t)}{dt} = -z(t)\beta + x(t)y(t)
\]

The standard values \(\sigma = 10, \rho = 28, \text{and } \beta = 8/3\) were used here and also in \[3\]. \(x(0), y(0), \text{and } z(0)\) were set to 1. The goal is to predict \(x(t + 1)\) based on \([x(t), x(t - 1), x(t - 2), x(t - 3)]\). Like in \[3\], 500 datapoints were generated to be used in our experiments.

Box-Jenkins Gas-Furnace (Gas) is a well known dataset. The problem is to predict \(CO_2\) concentration \(y(k)\) based on 4 previous concentration values \([y(k - 1), \ldots, y(k - 4)]\) and 6 previous methane feed rate values \([u(k - 1), \ldots, y(k - 6)]\). There are 296 datapoints in this dataset and it is available, for example, from the website of Greg Reinsel: http://www.stat.wisc.edu/~reinsel/bjr-data.

Fig. 2 Further optimization is performed using the original undiscretized data.
4.2 Experimental Setup

For all the datasets, the parameters in Table 1 were applied. C4.5 was used with its default parameters defined in [12] and the output was partitioned with 5 output MFs in all experiments. The number of generations \( G \) and population size \( N_{\text{pop}} \) for Electric problem were both 1000 (1,000,000 fitness evaluations). For MG, Lorenz and Gas datasets \( G = 350 \) and \( N_{\text{pop}} = 600 \) (210,000 fitness evaluations). With all datasets, 5-fold cross-validation was repeated 6 times (total 30 runs) with different random seeds.

<table>
<thead>
<tr>
<th>Distribution index for mutation</th>
<th>Distribution index for cross-over</th>
<th>Cross-over probability</th>
<th>Mutation probability</th>
<th>Merging threshold</th>
<th>( \Delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>20</td>
<td>0.9</td>
<td>( 1/\theta )</td>
<td>0.7</td>
<td></td>
</tr>
</tbody>
</table>

The results are reported by drawing the Pareto fronts, which can be different for each run. There can also be FMs having the same number of rules but different number of rule conditions. To present an average result over a run, the average of the rule conditions \( R_{\text{cond}} \), training accuracy \( \text{MSE}_{\text{trn}} \), and testing accuracy \( \text{MSE}_{\text{tst}} \) of FMs with certain number of rules are computed for each run. Then those values are averaged over the runs they were present. To obtain reliable results, it is required that an FM with certain number of rules is present in the Pareto front at least in 15 out of 30 runs in order to be included in the Figures.

4.3 MG, Lorenz and Gas Datasets: Results Comparison

Our proposal is compared to a MOEA-based approach [3], which starts with random initialization of the population. Rules and rule conditions are selected but MFs are not tuned. As optimization algorithms, several well known algorithms, such as, NSGA-II, PAES, (2+2)PAES, and MOGA, were used. However, the results of MOGA are excluded, because the root mean squared error (RMSE) obtained by it was clearly worse than the RMSE of the other methods. Including those results would have made depicting of Figures, in which results are compared, difficult.

In that paper, a Table containing the information of the first FMs having RMSE for train set lower than a defined threshold was presented. The first FMs means that they are the most compact FMs of the Pareto front having the RMSE less than the threshold. Those thresholds were selected, such that, they present a good RMSE.

---

2 Each dataset was divided into 5 subsets. Learning was performed with 4 subsets and testing with the remaining subset. The procedure was iterated 5 times, such that, each time a different subset was used for testing.
value for a particular problem. For MG, Lorenz, and Gas data sets the thresholds were 0.08, 1, and 0.6, respectively.

Our results, like mentioned in section 4.2, are the result of 5-fold cross-validation repeated 6 times. They are presented in Fig. 3. It is seen that for MG and Lorenz datasets, our FMs dominate the FMs of the comparative study. It is also noticed that for Gas data, none of our FMs is dominated by the FMs of the comparative study, which are dominated by many of our FMs.

### 4.4 Electric Data: Results Comparison

Our results were compared to another MOEA based Mamdani FM identification approach [2]. It creates the initial population using WM algorithm and uses MOEAs (NSGA-II [6] and SPEA2 [18]) to tune the MFs and to select rules (50 000 fitness evaluations). Note, that rule conditions are not selected by it but the whole rule is either included or not included into rule base. Thus each rule has as many rule conditions as there are input variables in the dataset.

The accuracy of the FMs was analyzed in [2] as follows. For each run, the FM with the lowest \( \text{MSE}_{\text{trn}} \) was selected and the number of rules \( R \), number of rule conditions \( R_{\text{cond}} \) and \( \text{MSE}_{\text{tst}} \) of that FM were recorded. They were averaged over the number of runs and standard deviations \( \sigma \) of \( \text{MSE}_{\text{trn}} \) and \( \text{MSE}_{\text{tst}} \) were reported. Moreover, T-test with 95% confidence was reported for the \( \text{MSE}_{\text{trn}} \) and \( \text{MSE}_{\text{tst}} \).

![Fig. 3 The results comparison for four datasets](image)
Here we perform exactly the same analysis and use the same notations as in [2]: ⋆ stands for the best averaged result in the column, + means that the performance of the corresponding row is worse than the best result, and = means that there is no significant difference compared to the best result.

From the results presented in Table 2 it is seen that the simplified initial FMs, denoted by C4.5, are more compact than the final FMs in comparative study. However, their accuracy need to be improved by learning process. It is seen that for both train and test sets, the T-test shows that our final FMs are significantly more accurate than final FMs in [2]. Yet, they contain far less rules and rule conditions than in comparative study. That can also be seen from Fig. 3, in which the Pareto front alongside with the results of [2] is shown.

<table>
<thead>
<tr>
<th>Reference Method</th>
<th>R</th>
<th>R_{cond}</th>
<th>MSE_{trn}/2</th>
<th>σ_{trn}</th>
<th>T-test</th>
<th>MSE_{tst}/2</th>
<th>σ_{tst}</th>
<th>T-test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alcalá et al. [2] WM+NSGA-II</td>
<td>41</td>
<td>164</td>
<td>14488</td>
<td>965</td>
<td>+</td>
<td>18419</td>
<td>3054</td>
<td>+</td>
</tr>
<tr>
<td>This paper C4.5+NSGA-II</td>
<td>13.4</td>
<td>23.8</td>
<td>11086</td>
<td>1993</td>
<td>*</td>
<td>13035</td>
<td>3734</td>
<td>*</td>
</tr>
</tbody>
</table>

5 Conclusions

We proposed a genetic fuzzy system (GFS) for identification of Mamdani fuzzy models (FMs). It simultaneously optimizes the parameters of fuzzy sets and selects rules and rule conditions. Learning process is not solely based on genetic operators but is aided by inconsistent rule removal. Besides, the initial population is created based on a modification to decision tree (DT) initialization proposed in [1].

Our method was compared to two genetic fuzzy systems [2, 3]. It was seen that usually our FMs contained less rules and rule conditions and were more accurate than the FMs of the comparative studies.

The focus of this paper was on obtaining accurate FMs with very small rule bases. Interpretability of fuzzy partition is not always guaranteed by our GFS and a future study is needed to consider that aspect as well.

References

Publication P4


© 2009 IEEE. Reprinted with permission.
A Dynamically Constrained Multiobjective Genetic Fuzzy System for Regression Problems

Pietari Pulkkinen and Hannu Koivisto

Abstract—In this paper, a multiobjective genetic fuzzy system (GFS) to learn the granularities of fuzzy partitions, tuning the membership functions (MFs), and learning the fuzzy rules is presented. It uses dynamic constraints, which enable three-parameter MF tuning to improve the accuracy while guaranteeing the transparency of fuzzy partitions. The fuzzy models (FMs) are initialized by a method that combines the benefits of Wang–Mendel (WM) and decision-tree algorithms. Thus, the initial FMs have less rules, rule conditions, and input variables than if WM initialization were to be used. Moreover, the fuzzy partitions of initial FMs are always transparent. Our approach is tested against recent multiobjective and monoobjective GFSs on six benchmark problems. It is concluded that the accuracy and interpretability of our FMs are always comparable or better than those in the comparative studies. Furthermore, on some benchmark problems, our approach clearly outperforms some comparative approaches. Suitability of our approach for higher dimensional problems is shown by studying three benchmark problems that have up to 21 input variables.

Index Terms—Genetic fuzzy systems (GFSs), initialization, accuracy, interpretability, Mamdani fuzzy models (FMs).

I. INTRODUCTION

INTERPRETABILITY-accuracy tradeoff of fuzzy models (FMs) has recently attained a lot of research interest [1]–[9]. Since it is not possible to maximize these contradicting objectives simultaneously, multiobjective evolutionary algorithms (MOEAs) have recently been used to find a Pareto optimal set of FMs that present different tradeoffs between the objectives. These approaches are also called multiobjective genetic fuzzy systems (GFS) [10], [11].

Accuracy is often measured by mean-squared error (MSE) when regression problems are considered. However, there is no exact measure for interpretability of FMs [2] and it tends to be somewhat subjective. Nevertheless, the definition by Ishibuchi and Yamamoto [12] is often used. It defines interpretability by four factors: 1) transparency of fuzzy partitions; 2) complexity of FMs (e.g., the number of fuzzy rules and input variables); 3) complexity of fuzzy-rule base (e.g., type of rules and the number of rule conditions); and 4) complexity of fuzzy reasoning (e.g., defuzzification method).

Factor 1) is often satisfied by using fixed fuzzy partitions (uniformly distributed or known by a priori knowledge) [3], [12]. However, a priori knowledge is often not available. Furthermore, if fuzzy partitions do not present the real distribution of data, the accuracy of FMs is deteriorated [13]. Thus, it is important to not only optimize the rules and rule conditions, but also the membership-function (MF) parameters. However, this increases the search space and may deteriorate the transparency of fuzzy partitions.

There are also studies in which fuzzy partitions are not fixed and factor 1) is taken into account by other means. Merging of highly similar fuzzy sets was used in [14] and [15] to improve the transparency of fuzzy partitions. Parameters of a fuzzy set that cover another fuzzy set were automatically adjusted in [4]. Penalties were issued in [5], if the intersection point of two fuzzy sets was not between user-specified boundaries. This approach not only avoided highly overlapping fuzzy sets, but also ensured that the whole universe of discourse (UoD) was strongly covered. The approach [5] was extended in [16] to reduce the effects of relaxed covering [4]. Here, [16] is followed; however, instead of minimizing the penalties, dynamic constraints are used to ensure that the fuzzy partitions are always transparent. This increases the selection pressure and improves the search efficiency [17].

This paper deals with regression (or function estimation) problems, which have not yet received as much research efforts as classification problems [6]. We apply Mamdani FMs [18], which are also called linguistic FMs. When regression problems are considered, the population is usually initialized randomly or by Wang and Mendel (WM) method [19]. Unfortunately, random initialization does not guarantee a good starting point for further optimization, and WM method usually leads to high number of rules and rule conditions when high-dimensional problems and/or problem with many data points are considered. Recently, we proposed a decision-tree (DT) based initialization method for regression problems [20], which reduces the number of input variables and leads to less rules and rule conditions than WM initialization. However, it does not necessarily create transparent fuzzy partitions. WM algorithm, on the other hand, creates rules for a priori given fuzzy partitions; thus, transparency of fuzzy partitions is usually high. Here, we combine the benefits of WM and DT initialization. Therefore, the initial fuzzy partitions are transparent, and the initial FMs contain less rules, rule conditions, and input variables than when WM algorithm is used.

The initial population is then optimized by multiobjective GFS that uses dynamic constraints to ensure the transparency of fuzzy partitions. It also reduces the number of rules, rule conditions, MFs, and input variables. The proposed initialization method and multiobjective GFS therefore aid to satisfy the aforementioned factors [1]–[3]). Factor 4), which is the complexity of fuzzy reasoning, is taken into account by applying simple-weighted-average-defuzzification method.
Our multiobjective GFS is tested on a set of nine benchmark problems having 2 up to 21 input variables. For six of them, there are results of other recently proposed GFSs available. Our results are compared to them, and it is shown that our results are comparable or better in terms of accuracy and interpretability.

This paper is organized as follows. First, a brief survey of recently proposed multiobjective GFSs is given. Based on this, novelty of our multiobjective GFS is clearly pointed out. Then, the interpretability of FMs is discussed and a special attention is paid to transparency of fuzzy partitions. Then, in Section IV, the proposed initialization method is introduced. After this, in Section V, dynamically constrained multiobjective GFS is presented. The results comparisons are performed in Section VI and conclusions are given in Section VII.

II. MULTIOBJECTIVE GENETIC FUZZY SYSTEMS FOR LINGUISTIC-LINEAR MODEL IDENTIFICATION: STATE OF THE ART

Recently, several researchers have focused on designing multiobjective GFSs to identify of compact and accurate linguistic FMs. Ishibuchi’s research group has published several papers that consider fuzzy classification. Nonetheless, until recently, there were hardly any papers that considered multiobjective GFSs in regression problems [23].

Table I presents multiobjective GFSs for classification and regression problems. For the sake of brevity, it includes only the recent approaches that apply the second-generation MOEAs (e.g., the nondominated sorting genetic algorithm II (NSGA-II), the strength pareto evolutionary algorithm 2 (SPEA2), and the pareto archived evolution strategy (PAES)). It also excludes those approaches that apply first-order Takagi–Sugeno FMs. In this table, rule selection means that a rule is either included or not included into an FM, whereas rule learning means that appropriate rule conditions are learned by GFS. It is seen that usually either rule learning or rule selection is applied, and there is only one approach [27] that applies neither of them.

MFs of fuzzy rules are taken from four different fuzzy partitions in [1], which means that the resulting global fuzzy partitions are not always transparent. Granularities of global fuzzy partitions are learnt in [24], which improves the transparency. The most trivial way to obtain transparent fuzzy partitions is to use evenly distributed uniformly shaped MFs, like in [3]. However, MFs tuning is often applied because it usually improves the accuracy. Unfortunately, it often deteriorates the transparency of fuzzy partitions. In the area of regression problems, there are some methods [21], [22], [25]–[27] that apply MFs tuning and have appropriately considered this factor. One of them [27] is a context-adaptation approach that only performs MFs tuning, requiring the whole rule base to be provided by the user. MF parameters are learnt using a linguistic two-tuple tuning scheme [9] in [21] and [22]. Piecewise-linear-transformation techniques are applied in [25] and a wrapper-based embedded process is used in [26]. The approaches [8], [20], [23] apply conventional three-parameter MFs tuning with static constraints, which does not guarantee transparency of fuzzy partitions.

In this paper, three-parameter MFs tuning with dynamic constraints is applied. The search space is therefore larger compared to two-tuple representation, which only modifies the lateral displacements of the MFs. On the other hand, it is excepted that the proposed approach improves the accuracy. Moreover, because of dynamic constraints, it is guaranteed that the whole UoD is strongly covered and there is no highly overlapping MFs. Our approach also does not require that MFs are uniformly shaped as long as the transparency conditions, which are introduced later in Section III-A, are met. In some cases, uniformly shaped MFs can actually be misleading if they do not present the real distribution of the data. In some cases, it is therefore necessary that some fuzzy sets are, for example, wide, whereas some others are narrow. Finally, granularities of global fuzzy partitions are also learnt by our approach. These properties guarantee that our approach maintains the transparency of fuzzy partitions at a good level.

Input-variable selection before applying GFS (i.e., in initialization phase) reduces the number of parameters to be optimized. This has been applied by some approaches; however, in the field of regression, there is only one approach [20] that applies this. Usually, regression problems with 2 up to 10 input variables are studied in the literature, and therefore, the role of input-variable selection is not crucial. However, in this paper, its role becomes more important as problems up to 21 input variables are studied.

The difference between the proposed approach and the approach [16] is more than just a different problem type. Transparency of fuzzy partitions was obtained in [16] by minimizing a transparency index. It means that the transparency indexes of FMs in population may be very different. There may be some
FM with highly transparent fuzzy partitions and some other FMs with unacceptable fuzzy partitions. Naturally, by constraining the range in which the value of transparency index can vary reduces the variation. However, in this case, the offspring population will usually contain some infeasible FMs (FMs for which the transparency index is not acceptable). This deteriorates the search efficiency of GFS. In this paper, transparency of fuzzy partitions is guaranteed by dynamic constraints. This reduces the number of fitness objectives by one, which increases the selection pressure [17].

Based on this brief analysis, it can be concluded that the proposed multiobjective GFS is novel. Indeed, to the best of our knowledge, there exist no multiobjective GFS applicable to regression, which performs rule learning and three-parameter MFs tuning, while preserving transparency of fuzzy partitions. Moreover, input variables are selected in two ways. First, during the initialization phase. Second, during the multiobjective-GFS-search process, which can select input variables among the remaining ones after the initialization.

III. INTERPRETABILITY OF FUZZY MODELS

As mentioned previously, in this paper, the factors 2) and 3) of the interpretability definition [12] are satisfied by minimizing the complexity of FMs and factor 4) by application of simple-weighted-average defuzzification. However, because the MFs are tuned, factor 1)—transparency of fuzzy partitions—requires a special attention. In the next section, a definition for this is given. It applies only to input variables, because in this paper, singleton output MFs are used. Because singleton MFs can be presented with only one parameter, it is sufficient to apply static constraints, introduced later in Section V-B, to maintain the transparency of output partition at a good level.

A. Transparency of Fuzzy Partitions

As in [27], this paper uses the transparency definition by de Oliveira [28], which states that a transparent fuzzy partition must meet the conditions, which are given as follows.

1) The number of MFs per variable is moderate.
2) MFs are distinguishable, i.e., two MFs do not present the same or almost the same linguistic meaning.
3) Each MF is normal. An MF is normal if it has membership value 1 at least at one point of UoD.
4) UoD is strongly covered. At least one MF receives a membership value $\beta$ (where $\beta > 0$) at any point of UoD.

Condition 1) is easily met by constraining the maximum number of MFs to a moderate number (for example, 9). Also, condition 3) is met by applying normal MFs and genetic operators that do not alter their normality. Meeting conditions 2) and 4) is more challenging. In this paper, it is considered that they are met if globally defined MFs are used and the following conditions are met.

1) Symmetry condition: The shapes of all MFs are symmetrical. For example, Gaussian MF and generalized-bell (gbell) MF are symmetrical by definition. Also, other MF types, such as triangular and trapezoidal MFs can be easily made symmetrical.

2) $\alpha$-condition: At any intersection point of two MFs, the membership value is at most $\alpha$.
3) $\gamma$-condition: At the center of each MF, no other MF receives membership value larger than $\gamma$. Center of an MF depends on which MF type is used. For gbell MF (with parameters $a$, $b$, and $c$) and Gaussian MF (with parameters $c$ and $\sigma$), center is the parameter $c$. For triangular MF (with parameters $a < b < c$), $b$ is the center. For trapezoidal MF (with parameters $a < b < c < d$), center is $b + ((c - b)/2)$ (see also Fig. 1).
4) $\beta$-condition: UoD is strongly covered, i.e., at each point of UoD, at least one MF has membership value at least $\beta$.

Fig. 1 shows examples of fuzzy partitions with settings $\beta = 0.05$, $\gamma = 0.25$, and $\alpha = 0.8$. Section III-B describes how $\beta$, $\gamma$, and $\alpha$ must generally be selected in order to apply the dynamic-tuning strategy.

In this paper, gbell MFs are used. They are defined as

$$
\mu(x; a, b, c) = \frac{1}{1 + |(x - c)/a|^{2b}}
$$

where $a$, $b$, and $c$ define the width, shape, and center of an MF, respectively. As gbell MFs are symmetrical, first of the previous conditions is met. Fulfillment of the rest three conditions rely largely on computing the values of $x$, for which an MF receives a certain membership value $\mu$. Because of the symmetry of gbell MFs, any membership value $\mu \in (0, 1)$ is received on the left and right side of the center $c$. These points are denoted here by $I_L$ and $I_R$

$$
I_L(\mu, p) = c - a (\kappa(\mu))^{1/2b}, \quad \mu \in (0, 1)
$$

$$
I_R(\mu, p) = c + a (\kappa(\mu))^{1/2b}, \quad \mu \in (0, 1)
$$

where $p = [a, b, c]^T$ is a vector containing the MF parameters and

$$
\kappa(\mu) = \frac{1 - \mu}{\mu}, \quad \mu \in (0, 1).
$$

Authorized licensed use limited to: Tampereen Teknillinen Korkeakoulu. Downloaded on February 17,2010 at 04:50:58 EST from IEEE Xplore. Restrictions apply.
Equations (2) and (3) are used to formulate the \( \alpha, \gamma, \) and \( \beta \) conditions. For the sake of clarity, each of them is split into two parts, denoted here by left and right. They ensure the fulfillment of the conditions on the left or right side of the center of an MF, respectively. Let the active MFs of a variable be indexed as \( j = 1, \ldots, M_A \), where \( M_A \) is the number of currently active MFs of that variable. It will be shown later that our multiobjective GFS maintains the ordering of MFs, i.e., if \( i > j \), then \( c_i > c_j \), where \( c_i \) and \( c_j \) are the gbell parameters of MFs \( i \) and \( j \). Moreover, in this paper, the fuzzy partitions with only one MF are not allowed, because they are not considered transparent. Hence, throughout this paper, it is known that if \( j = 1 \), then MF is the leftmost MF and its neighboring MF is \( j + 1 \). If \( 1 < j < M_A \), then MF is in the middle of neighboring MFs \( j - 1 \) and \( j + 1 \). Finally, if \( j = M_A \), then MF is the rightmost MF and the neighboring MF is \( j - 1 \). Thus, the transparency conditions can be written as follows:

**Right \( \alpha \)-condition**

\[
I_R(\alpha, p_j) \leq I_L(\alpha, p_{j+1}), \quad \text{if } j < M_A.
\]

**Left \( \alpha \)-condition**

\[
I_L(\alpha, p_j) \geq I_R(\alpha, p_{j-1}), \quad \text{if } j > 1.
\]

**Right \( \gamma \)-condition**

\[
I_R(\gamma, p_j) \leq c_{j+1} \land c_j \leq I_L(\gamma, p_{j+1}), \quad \text{if } j < M_A.
\]

**Left \( \gamma \)-condition**

\[
I_L(\gamma, p_j) \geq c_{j-1} \land c_j \geq I_R(\gamma, p_{j-1}), \quad \text{if } j > 1.
\]

**Right \( \beta \)-condition:**

\[
\begin{cases}
I_R(\beta, p_j) \geq I_L(\beta, p_{j+1}), & \text{if } j < M_A \\
I_R(\beta, p_j) \geq \chi_{\text{high}}, & \text{if } j = M_A
\end{cases}
\]

**Left \( \beta \)-condition:**

\[
\begin{cases}
I_L(\beta, p_j) \leq I_R(\beta, p_{j-1}), & \text{if } j > 1 \\
I_L(\beta, p_j) \leq \chi_{\text{low}}, & \text{if } j = 1
\end{cases}
\]

where the range variable is \( \chi = \chi_{\text{high}} - \chi_{\text{low}} \), where \( \chi_{\text{low}} \) and \( \chi_{\text{high}} \) are the lower and upper bounds of the variable, respectively. These conditions are the basis of the proposed dynamic constraints, which require that the fuzzy partitions of initial FMs are transparent. Thus, two simple partition algorithms to create transparent fuzzy partitions are introduced next.

**B. Partitioning Algorithm to Create Evenly Distributed Fuzzy Partition**

This algorithm creates a fuzzy partition consisting of \( M_A \) evenly distributed uniformly shaped MFs, and it is only used when creating the first FM of the initial population. Because MFs are uniformly shaped, the gbell parameter \( a \) for each MF \( j \) is

\[
a_j = a_{\text{even}} = \frac{\chi}{2(M_A - 1)}, \quad j = 1, \ldots, M_A.
\]

It is required that each \( a_j \geq a_{\text{min}} = 0.025\chi \) to avoid very narrow MFs. This limits the maximum value of \( M_A \) to 21; however, in practice, more than nine MFs are hardly ever assigned. The minimum value of \( M_A \) is 2. Centers are distributed evenly as

\[
c_1 = \chi_{\text{low}}, \quad \text{and} \quad c_j = c_{j-1} + \frac{\chi}{M_A - 1}, \quad j = 2, \ldots, M_A.
\]

Assigning the values for \( a \) and \( c \) according to (5) and (6) guarantees that UoD is strongly covered and the membership value of each MF pair at their intersection point is 0.5. Thus, \( 0 < \beta < 0.5 \) and \( 0.5 < \alpha < 1 \) must be selected in order to apply the dynamic-tuning strategy. Because the membership value at each intersection point is 0.5, the \( \beta \) and \( \alpha \) conditions are fulfilled. Moreover, because gbell MFs are symmetrical, the symmetry condition is satisfied as well. The \( \gamma \)-condition requires that at the center of each MF, no other MF receives membership value larger than \( \gamma \). This algorithm selects \( b \), such that, at the center of each MF, the neighboring MF(s) receive the membership value \( \gamma^* = 0.05 \). Thus, \( \gamma^* < \gamma < 0.5 \) must be selected in order to apply the dynamic-tuning strategy. The following formula for selecting \( b \) can be derived by starting from either (2) or (3):

\[
b_j = \frac{\ln k(\gamma^*)}{2\ln(d_{\text{center},j}/a_j)}, \quad j = 1, \ldots, M_A
\]

where

\[
d_{\text{center},j} = \begin{cases}
\min(c_j - c_{j-1}, c_{j+1} - c_j), & \text{if } 1 < j < M_A \\
c_{j+1} - c_j, & \text{if } j = 1 \\
c_j - c_{j-1}, & \text{if } j = M_A
\end{cases}
\]

denotes the minimum distance from \( c_j \) to the nearest center(s) of neighboring MF(s).

Because MFs are evenly distributed, \( d_{\text{center},j} = \chi/(M_A - 1) \forall j \). Thus, (7) can be written as

\[
b_j = \frac{\ln k(\gamma^*)}{\ln 4}, \quad j = 1, \ldots, M_A.
\]

There is no upper limit for the value of \( b \) in the sense that larger \( b \) values will not violate the transparency conditions. However, very large \( b \) values are not desired as they make gbell MFs similar to crisp sets and because \( b \) is the exponent in (1). Therefore, value of \( b \) for each MF is defined by (9) by this algorithm.

**C. Partitioning Algorithm to Create Unevenly Distributed Fuzzy Partition**

As there is no \textit{a priori} knowledge about the distribution of MFs, it is also beneficial to create unevenly distributed nonuniformly shaped MFs. The following algorithm is used for this purpose, and it is applied to create the fuzzy partitions of the rest FMs of the initial population and as a part of genetic operators. It selects \( c \) and \( a \) as follows:

\[
a_1 = \max(a_{\text{min}}, r_1a_{\text{even}}), \quad \text{and} \quad c_1 = \chi_{\text{low}}
\]

\[
a_j = \max\left(a_{\text{min}}, r_j\left(\frac{(2j-1)a_{\text{even}} - (c_{j-1} + a_{j-1})}{2}\right)\right)
\]

where \( j = 2, \ldots, M_A - 1 \)

\[
c_j = c_{j-1} + a_{j-1} + a_j, \quad j = 2, \ldots, M_A - 1
\]
where \( r_1, r_2, \ldots, r_{M_A-1} \in [0, 1] \) are random real numbers; \( a_{\text{even}} \) and \( a_{\text{min}} \) were defined in the previous section. It can be easily verified that by selecting \( r_1 = r_2 = \cdots = r_{M_A-1} = 1 \), this algorithm is identical to the algorithm in the previous section.

Unlike in the previous partition algorithm, here, parameter \( b \) values are randomly selected from interval \([1, 10]\). However, they are not allowed to be less than the corresponding minimum values computed according to (7). Thus, it is guaranteed that at the center of each MF, the neighboring MF(s) receive the membership value less than or equal to \( \gamma^* \).

It is seen from (10), (11), and (13) that the more narrow MFs are more likely to be located on the left side of the range and the wider MFs on the right side of the range. There is, naturally, no justification for this. Hence, by uniform chance, the parameters are defined either by (10)–(13) or by their inversion as follows:

\[
\begin{align*}
a^*_j &= a_{M_A-j+1}, & b^*_j &= b_{M_A-j+1}, & c^*_j &= \chi_{\text{high}} - c_{M_A-j+1} \\
\end{align*}
\]

(14)

where \( j = 1, \ldots, M_A \).

As an example, consider creating a fuzzy partition with five MFs in range \([0, 1]\). From (5), it follows that \( a_{\text{even}} = 1/8 \). Let \( r_1 = 1/2, r_2 = 1, r_3 = 1/2, \) and \( r_4 = 1/2 \). Thus, \( a_1 = a^*_1 = 1/16, c_1 = 0, a_2 = a^*_3 = 5/32, c_2 = 7/32, a_3 = a^*_1 = 1/16, c_3 = 7/16, a_4 = a^*_2 = 3/32, c_4 = 19/32, a_5 = a^*_5 = 5/16, \) and \( c_5 = 0, c^*_1 = 3/32, c^*_2 = 9/16, c^*_3 = 25/32, \) and \( c^*_5 = 1 \). The minimum values for \( b_1 = b^*_5, b_2 = b^*_1, b_3 = b^*_3, b_4 = b^*_2, \) and \( b_5 = b^*_2 \), according to (7), are 1.1752, 4.3755, 1.6067, 2.8820, and 5.6114, respectively. Fig 2(a) shows the resulting partition when centers are computed according to (10)–(13), whereas Fig 2(b) depicts the resulting partition when (14) is used instead. It is seen that although MFs are nonuniformly shaped and unevenly distributed, the fuzzy partitions are transparent and reasonable linguistic values could be given. In Fig 2, \( \beta = 0.05, \gamma = 0.25, \) and \( \alpha = 0.8 \).

**Fig. 2.** Example of (a) unevenly distributed fuzzy partition and (b) its inverse.

\[
a_{M_A} = \chi_{\text{high}} - (c_{M_A-1} + a_{M_A-1}), \quad \text{and} \quad c_{M_A} = \chi_{\text{high}}
\]

(13)

**IV. POPULATION INITIALIZATION**

Whenever a GFS is used, the population needs to be initialized first. In order to reduce the search space, it is desirable that the initialization method is able to select the relevant input variables. Thus, in [15], the C4.5 [29] DT-based method for classification problems was proposed. Recently, in [20], it was made suitable for regression problems. Although this method is capable of selecting relevant input variables, its main limitations are that: 1) it does not guarantee transparent fuzzy partitions and 2) it may create far more rules than necessary when applied to noisy datasets.

In this paper, DT initialization is neither used to create the rule base nor to initialize MF parameters, but to select relevant input variables, to reduce the number of input MFs, and rule conditions. MF parameters are determined by the introduced partition algorithms (see Section III-B and Section III-C), which guarantee transparency of fuzzy partitions. Rule base is created by slightly modified WM algorithm [19]. The proposed two modifications are that: 1) when a data point is matched to MFs in order to generate a rule, the data point is not always matched to MFs of all possible input variables. Instead, it is first classified by the constructed DT, and only those input variables that were used by DT to classify the data point are used for matching and 2) as WM algorithm may create large number of rules for datasets with many data points and/or input variables, the generated rules are divided among the members of initial population and only a portion of them is allowed to be included into one FM.

**A. Creation of the First Fuzzy Method of the Population**

The procedure of creating the first FM is shown in Fig. 3. It is started by discretizing the continuous output data in order to apply C4.5 algorithm. This is done by dividing the output to \( M_{\text{out}} \) crisp regions. Each continuous output value falls into one of these \( M_{\text{out}} \) regions and it is replaced with corresponding class label \( S \in \{1, \ldots, M_{\text{out}}\} \), which represents these regions. Then, C4.5 algorithm can be applied and a DT constructed.

All input variables which are not used by DT are then removed. Then, fuzzy partitions for the remaining input variables and for the output are created. A user is required to provide the number of input MFs \( M_{in} \) and the number of output MFs \( M_{out} \). However, the DT can be used to limit the number of
input MFs. First, the DT is transformed into an FM, according to [15]. After this, the number of MFs for each input variable \( j \) in the resulting FM is checked and denoted by \( M_{\text{DT},j} \). Then, instead of partitioning each input variable with \( M_{\text{int}} \) MFs, each input partition is created with \( \min(M_{\text{DT},j}, M_{\text{int}}) \) MFs. The output is partitioned with \( M_{\text{out}} \) MFs. These partitions consist of uniformly shaped evenly distributed MFs and are created by the algorithm introduced in Section III-B.

Then, a slightly modified WM algorithm is used. As mentioned previously, when a rule is generated, each data point is first classified by the constructed DT and only those input variables that were used by DT to classify the data point are used for matching and become conditions of the generated rule. All other parts of the classical WM algorithm remain unchanged.

After creating the rule base, the number of active MFs \( M_{\text{A},j} \) for each input variable \( j \) is checked (an MF is active if it is part of at least one of the rules). If \( M_{\text{A},j} < \min(M_{\text{DT},j}, M_{\text{int}}) \), then there is a gap in fuzzy partition and the whole UoD is not strongly covered. If this is the case and if \( M_{\text{A},j} \geq 2 \), then a new evenly distributed partition with \( M_{\text{A},j} \) MFs is created. If \( M_{\text{A},j} < 2 \), then input variable \( j \) is removed and \( M_{\text{A},j} \) is set to 0. The maximum number of MFs, i.e., \( M_{\text{max}} = M_{\text{A},j} \), that each FM of the population can use in input variable \( j \) is determined by this phase. Also, all the input variables that are not removed until now form the set of candidate input variables. The number of these remaining input variables is denoted by \( n_s \).

The generated rule base may contain large amount of rules. However, in this paper, each FM can contain at most \( R_{\text{max}} = 30 \) rules. If the rule base has more than \( R_{\text{max}} \) rules, then \( R_{\text{max}} \) rules are randomly selected out of it. Otherwise, the rule base is taken as a whole. If rules are randomly selected, it may result into some gaps in the fuzzy partition, which is not allowed. In this case, it is required that the number of active MFs for each input variable must be \( M_{\text{out}} \), and the number of active output MFs must be \( M_{\text{out}} \). If this is not the case, then \( \max(M_{\text{max}1}, M_{\text{max}2}, \ldots, M_{\text{max}s}, M_{\text{out}}) \) randomly selected rules are replaced with some rules, thus making all the inactive MFs active. In this paper, these rules are created, such that, in the first of the rules, all antecedents and the consequent are 1. In the second rule, they are all 2. This is continued until all inactive MFs have become active. Of course, it must be taken care that the antecedents for input variable \( j \) are at most \( M_{\text{max}} \), and, for the consequent, at most \( M_{\text{out}} \). This rule replacement is necessary only if the rule base contains more than \( R_{\text{max}} \) rules. Otherwise, it is certain that there are no gaps in the fuzzy partition.

B. Mamdani Fuzzy Model and Its Coding for Multiobjective-Genetic-Fuzzy-System Optimization

The original dataset contains \( n \) input variables; however, the initialization method selects \( n_s \leq n \) of them. Therefore, a dataset with \( D \) data points is denoted as \( Z = [X \ y] \), where \( X \) is \( D \times n_s \) input matrix, and \( y \) is \( D \times 1 \) output vector. The first FM and all other FMs in this paper are Mamdani FMs. Mamdani fuzzy rules are expressed as

\[
R_i : \text{if } x_1 \text{ is } B_{i,1} \ldots \text{ and } x_n \text{ is } B_{i,n}, \text{ then } C_i
\]

where \( B_{i,j} \), with \( j = 1, \ldots, n_s \) and \( i = 1, \ldots, R \), is an input fuzzy set, \( C_i \) is an output fuzzy set, and \( R \) is the number of rules. To reduce the computational costs, the output of FMs is computed by approximation of centroid of gravity method [3], [30] as

\[
\hat{y}_k = \frac{\sum_{i=1}^{R} \beta_i(x_k) \tilde{C}_i}{\sum_{i=1}^{R} \beta_i(x_k)}, \quad k = 1, \ldots, D \tag{15}
\]

where \( C_i \) is the center value of \( C_i \), and \( \beta_i(x_k) = \prod_{j=1}^{n_s} B_{i,j}(x_k,j) \) is the degree of rule activation. When the slightly modified WM algorithm was used to create the rule base, gbell output MFs were used. However, at the optimization phase, application of gbell MFs is not necessary anymore, since \( C \) is the only output FM parameter affecting the outcome. Therefore, all gbell output MFs are replaced with singleton MFs as

\[
\mu(x, C) = \begin{cases} 
1, & \text{if } x = C \\
0, & \text{if } x \neq C
\end{cases}
\]

where \( C \) is the corresponding gbell MF parameter. For the purpose of multiobjective GFS optimization, the antecedents of the rule base are presented with an integer-coded matrix \( A \). It specifies for each rule \( i = 1, \ldots, R \) that MF is used for input variable \( j = 1, \ldots, n_s \)

\[
A = \begin{bmatrix}
    a_{1,1} & a_{1,2} & \ldots & a_{1,n_s} \\
    a_{2,1} & a_{2,2} & \ldots & a_{2,n_s} \\
        \vdots & \vdots & \ddots & \vdots \\
    a_{R,1} & a_{R,2} & \ldots & a_{R,n_s}
\end{bmatrix}
\tag{16}
\]

\( a_{i,j} \in \{0, 1, \ldots, M_{\text{max}}\} \), where \( M_{\text{max}} \) is the maximum number MFs in input variable \( j \). If \( a_{i,j} = 0 \), input variable \( j \) is not used in rule \( i \). Input variable \( j \) is not used in an FM if \( \forall i, a_{i,j} = 0 \), and rule \( i \) is not used in an FM if \( \forall j, a_{i,j} = 0 \). Input MF parameters to which each \( a_{i,j} \) is referring are defined in a real-coded matrix \( P \) as

\[
P = \begin{bmatrix}
    p_{1,1} & p_{1,2} & \ldots & p_{1,\rho} \\
    p_{2,1} & p_{2,2} & \ldots & p_{2,\rho} \\
        \vdots & \vdots & \ddots & \vdots \\
    p_{\rho,1} & p_{\rho,2} & \ldots & p_{\rho,\rho}
\end{bmatrix}
\tag{17}
\]

where \( \rho \) is the number of parameters used to define an MF. In this paper, \( \rho = 3 \), because gbell MFs are used. The maximum number of MFs in an FM is denoted by \( \delta = \sum_{j=1}^{n_s} M_{\text{max}j} \). Thus, for any \( a_{i,j} \neq 0 \), the corresponding input MF parameters are \( p_{1,j}, p_{2,j}, \ldots, p_{3,j} \), and \( p_{3,\rho} \), where

\[
l = \begin{cases} 
    a_{i,j}, & \text{if } j = 1 \\
    a_{i,j} + \sum_{k=1}^{j-1} M_k, & \text{if } j > 1
\end{cases}
\tag{18}
\]

Similarly as \( A \) states the input MFs used in the rules, an integer-coded vector \( s \) defines the output MFs (singleton) used in the rules. Formally, \( s = [s_1, s_2, \ldots, s_R]^T \), where \( s_i \in \{1, \ldots, M_{\text{out}}\} \), with \( i = 1, \ldots, R \). The maximum number of output MFs is denoted by \( M_{\text{out}} \). The output MF parameters to
which each $s_j$ is referring are defined in a real-coded vector $\mathbf{o} = [o_1, o_2, \ldots, o_{M_{\text{rest}}}]^T$. The total number of parameters to be optimized by a multiobjective GFS is $\theta = R_{n_s} + \rho \delta + R + M_{\text{out}}$, i.e., the sum of the cardinalities of $A$, $P$, $s$, and $o$.

### C. Mamdani-Fuzzy-Model Coding: An Example

Let us assume that the first FM of the initial population has four rules and uses two input variables $x_1$ and $x_2$, which are partitioned, respectively, with three and two gbell MFs. The output is partitioned with four singleton MFs. Both input variables and the output are in the range of $[0, 1]$. The partitions are uniformly shaped and evenly distributed. The rule base is given as follows.

- **Rule 1:** If $x_1$ is 1 and $x_2$ is 1, then output is 1.
- **Rule 2:** If $x_1$ is 2 and $x_2$ is 2, then output is 2.
- **Rule 3:** If $x_1$ is 2 and $x_2$ is 1, then output is 3.
- **Rule 4:** If $x_1$ is 3 and $x_2$ is 2, then output is 4.

This FM is coded as

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 2 & 1 \\ 3 & 2 \end{bmatrix}, \quad s = \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix}, \quad o = \begin{bmatrix} 0 \\ 1/3 \\ 2/3 \end{bmatrix}$$

$$P = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.5 & 0.5 \\ 2.124 & 2.124 & 2.124 & 2.124 & 2.124 \\ 0 & 0.5 & 1 & 0 & 1 \end{bmatrix}$$

where the first, second, and third row of $P$ contain the gbell parameters $a$, $b$, and $c$, respectively. The first three columns of $P$ contain the gbell parameters of the three MFs of $x_1$ and the rest two columns contain the gbell parameters of the two MFs of $x_2$. These parameters are computed according to the algorithm in Section III-B.

### D. Creation of the Rest of the Population

The first FM defines the maximum number of rules, maximum number of input variables, and maximum number of MFs per input variable for all the rest $N_{\text{pop}} - 1$ FMs of the population, where $N_{\text{pop}}$ is the population size.

If the rule base generated by slightly modified WM algorithm has more than $R_{\text{max}}$ rules, it means that some of the randomly selected rules in the first FM were replaced in order to avoid gaps in the fuzzy partition. In this case, one of the $N_{\text{pop}} - 1$ FMs receives the rule base (i.e., $A$ and $s$) of the first FM without any replacements. Then, $A$ and $s$ of the rest $N_{\text{pop}} - 2$ FMs are created by randomly selecting $R_{\text{max}}$ rules from the generated rule base.

If the generated rule base has at most $R_{\text{max}}$ rules, then the rule conditions $A$ of $N_{\text{pop}} - 1$ FMs are created by modifying the rule conditions of the first FM by replacing them with random conditions [7]. However, do-not-care conditions (i.e., conditions that are 0) are not allowed here, as it was pointed out in [8] that it is easier to obtain compact than accurate FMs. Rule consequences $s$ for all $N_{\text{pop}} - 1$ FMs are the same as in the first FM.

After creating $A$ and $s$ of the rest $N_{\text{pop}} - 1$ FMs, the input MF parameters $P$ are assigned based on $A$ of each individual FM. For each input variable $j$ of each FM, the number of active MFs $M_{A,j}$ is first checked. If $M_{A,j} \geq 2$, then a new unevenly distributed fuzzy partition with $M_{A,j}$ MFs is created by using the algorithm in Section III-C. If $M_{A,j} < 2$, then all nonzero rule conditions, if any, of that input variable are forced to zero. After this, the input variable has no active MFs, and the value of MF parameters for this input variable can be assigned to any value. However, if the genetic operators at a later stage cause at least two MFs to be active, then the value of these parameters is determined by the algorithm in Section III-C. Finally, the output MF parameters $o$ for all $N_{\text{pop}} - 1$ FMs are the same as in the first FM.

### E. Creation of the Rest of the Population: An Example

Let us return to the example from Section IV-C and consider creating one of the rest $N_{\text{pop}} - 1$ FMs. Since the initial FM has only $4 \leq R_{\text{max}}$ rules, the rules are created by modifying the rules of the first FM. Assume that as a result, the condition $If \ x_1 \ is \ 1 \ of \ the \ first \ rule \ was \ changed \ to \ If \ x_1 \ is \ 3$. Now, the FM has no rule in which the condition $If \ x_1 \ is \ 1$ is part of. Thus, the input MF 1 of $x_1$ is inactive and a new unevenly distributed partition is created with two MFs and assigned to input MFs 2 and 3 of $x_1$, such that their order is maintained. Similarly, a new unevenly distributed partition with two MFs is also created for $x_2$, which still has two active MFs. The following could be the result after these operations:

$$A = \begin{bmatrix} 3 & 1 \\ 2 & 2 \\ 2 & 1 \\ 3 & 2 \end{bmatrix}, \quad o = \begin{bmatrix} 0 \\ 1/3 \\ 2/3 \end{bmatrix}$$

$$P = \begin{bmatrix} 0.25 & 0.7 & 0.8 & 0.2 \\ 2.124 & 3 & 7 & 4 \\ 0 & 0.5 & 1 & 0 \end{bmatrix}$$

where the operated parameters are indicated with boldface. The parameter values of input MF 1 in $x_1$ are indicated with italics, because they are currently not important. If at some point of optimization, MF 1 becomes active again, the values will be assigned by the algorithm in Section III-C. Before this, none of the genetic operators will operate on these parameters.

### V. DYNAMICALLY CONSTRAINED MULTIOBJECTIVE GENETIC FUZZY SYSTEM

After the initialization, the further optimization is performed by popular NSGA-II [31]. Other parts of the algorithm are left unchanged; however, the original genetic operators are replaced with operators applying dynamic constraints, thus ensuring transparency of fuzzy partitions.

### A. Fitness Objectives

Two objectives to be minimized, which are as follows.

1. $MSE = (1/2D) \sum_{k=1}^{D} (y_k - \hat{y}_k)^2$, where $y_k$ and $\hat{y}_k$ are the actual and predicted outputs for data point $k$, and $D$ is the number of data points. This objective is actually $MSE/2$, but it is denoted here as $MSE$, which is quite common in the field of GFSs.
2) Number of active-rule conditions (total rule length): $R_{cond}$

The MSE objective is constrained, such that, each FM need to have $\text{MSE} \leq 1.5 \times \text{MSE}_{\text{initial}}$, where $\text{MSE}_{\text{initial}}$ is the MSE of the first FM of the initial population created in Section IV-A. This constraint is fairly easy to meet as it will be seen in Section VI that the accuracy can be significantly improved by multiobjective GFS optimization. However, it guarantees that the population does not contain some FMs only because they are very compact. Their accuracy must be reasonable as well.

B. Static Constraints for Output Membership Functions

As singleton output MFs are used, there is only one parameter to be optimized (lateral displacement). Therefore, they can be constrained by allowing them to slightly move left/right from their initial positions. The applied static constraints for output MF parameters are

$$\chi_{\text{low}} - \frac{\chi}{M_{\text{out}} - 1} \leq a_i \leq \chi_{\text{low}} + \frac{\chi}{2(M_{\text{out}} - 1)}$$

$$\chi_{\text{low}} + \frac{(2k - 3)\chi}{2(M_{\text{out}} - 1)} \leq a_k \leq \chi_{\text{low}} + \frac{(2k - 1)\chi}{2(M_{\text{out}} - 1)}$$

where $k = 2, \ldots, M_{\text{out}} - 1$

$$\chi_{\text{high}} - \frac{\chi}{2(M_{\text{out}} - 1)} \leq a_{M_{\text{out}}} \leq \chi_{\text{high}} + \frac{\chi}{M_{\text{out}} - 1}.$$  

This way of tuning resembles lateral-tuning method [9], and it guarantees transparency of output fuzzy partition to a good level.

C. Dynamic Constraints to Ensure the Transparency of Input Fuzzy Partition

This section presents the dynamic constraints guaranteeing transparent input fuzzy partition in case that a parameter of an MF is modified. The genetic operators assuring transparent input fuzzy partition in case that the number of MFs is altered are introduced later in Section V-D. A prerequisite for these dynamic constraints is that initially (i.e., before modification) the input fuzzy partition is transparent. This is guaranteed by the two partition algorithms, which have already been introduced in Section III-B and Section III-C.

MF parameters are modified one at a time. After each modification, the resulting fuzzy partition must satisfy the transparency conditions defined in Section III-A. As the initial fuzzy partitions are created by the algorithms in Section III-B and III-C, the ordering of MFs is initially known. The ordering is also known after each modification, because the dynamic constraints and the genetic operators in Section V-D do not allow to change it. Therefore, for any two MFs with parameters $a_i, b_i$, and $c_i$, and $a_j, b_j$, and $c_j$, where $j, i \in [1, M_A]$, with $i \neq j$, of an input variable that currently has $M_A$ active MFs, it is guaranteed that if $i > j$, then $c_i > c_j$, and vice versa. This is beneficial to design the dynamic constraints.

Besides the dynamic constraints, some static constraints also need to be satisfied; $a_j \in [a_{\text{low}}, a_{\text{high}}]$, where $a_{\text{low}} = 0.005\chi$, and $a_{\text{high}} = \chi$. Furthermore, $b_j \geq b_{\text{low}} = 1$, and it is preferred that $b_j \leq b_{\text{high}} = 10$; however, due to partition algorithms, it may be that initially $b_j > b_{\text{high}}$. In this case, $b_j$ is not allowed to increase anymore. Finally, $c_j \in [c_{\text{low}}, c_{\text{high}}]$, where $c_{\text{low}} = \chi_{\text{low}}$, and $c_{\text{high}} = \chi_{\text{high}}$. Next, the dynamic constraints are introduced. They can all be derived starting from (2) and (3).

1) Dynamic Constraints for Parameter $a$: If $a_j$ is increased (i.e., $MF_j$ becomes wider), the upper limit satisfying the $\gamma$-condition is

$$a_{\gamma,j} = \frac{d_{\text{center},j}}{(\kappa(\gamma))^{1/2b_j}}$$

where $\kappa(\gamma)$ and $d_{\text{center},j}$ are computed according to (4) and (8), respectively.

The upper limit satisfying the $\alpha$-condition is

$$a_{\alpha,j} = \frac{d_{\alpha,j}}{(\kappa(\alpha))^{1/2b_j}}$$

where

$$d_{\alpha,j} = \begin{cases} \min(I_L(\alpha, \beta_{j+1}) - c_j, c_j - I_R(\alpha, \beta_{j-1})), & \text{if } 1 < j < M_A \\ c_j - I_R(\alpha, \beta_{j-1}), & \text{if } j = M_A \\ I_L(\alpha, \beta_{j+1}) - c_j, & \text{if } j = 1 \end{cases}$$

is the minimum distance from $c_j$ to the point in which a neighboring MF receives membership value $\alpha$. $I_L$ and $I_R$ are computed according to (2) and (3), respectively.

If $a_j$ is decreased (i.e., $MF_j$ becomes more narrow), the lower limit satisfying the $\beta$-condition is

$$a_{\beta,j} = \begin{cases} \frac{d_{\beta,j}}{(\kappa(\beta))^{1/2b_j}}, & \text{if } d_{\beta,j} > 0 \\ a_{\text{low}}, & \text{if } d_{\beta,j} \leq 0 \end{cases}$$

where

$$d_{\beta,j} = \begin{cases} \max(I_L(\beta, \beta_{j+1}) - c_j, c_j - I_R(\beta, \beta_{j-1})), & \text{if } 1 < j < M_A \\ \max(\chi_{\text{high}} - c_j, c_j - I_R(\beta, \beta_{j-1})), & \text{if } j = M_A \\ \max(c_j - \chi_{\text{low}}, I_L(\beta, \beta_{j+1}) - c_j), & \text{if } j = 1 \end{cases}$$

is computed depending on the location of $MF_j$. If $d_{\beta,j} \leq 0$, UoD will be strongly covered regardless of the decrement in the value of $a_j$. In this case, the lower limit satisfying the $\beta$-condition is simply the static constraint $a_{\text{low}}$.

Combining the constraints yields to

$$\max(a_{\text{low}}, a_{\beta,j}) \leq a_j \leq \min(a_{\gamma,j}, a_{\alpha,j}, a_{\text{high}}).$$

2) Dynamic Constraints for Parameter $b$: If $b_j$ is increased (i.e., $MF_j$ becomes crisper), the following upper limit guarantees the fulfillment of $\alpha$-condition:

$$b_{\alpha,j} = \frac{\ln(\kappa(\alpha))}{2 \ln(\frac{d_{\alpha,j}}{a_j})},$$

if $d_{\alpha,j} < a_j$

$$b_{\text{high}},$$

if $d_{\alpha,j} \geq a_j$.

If $d_{\alpha,j} \geq a_j$, $MF_j$ receives at most a membership value $\alpha$ at any intersection point, regardless of the increment in the value of $b_j$. In this case, the upper limit is the static constraint $b_{\text{high}}$.\"
The following upper limit guarantees the fulfillment of $\beta$-condition:

$$b_{\beta,j} = \begin{cases} \frac{\ln(\kappa(\beta))}{2\ln(d_{\beta,j}/a_j)}, & \text{if } d_{\beta,j} > a_j \\ b_{\text{high}}, & \text{if } d_{\beta,j} \leq a_j. \end{cases}$$

If $d_{\beta,j} \leq a_j$, MF $j$ receives at least a membership value $\beta$ at the intersection point(s) of its neighboring MF(s), which is regardless of the increment in $b_j$. In this case, the upper limit is the static constraint $b_{\text{high}}$.

If $b_j$ is decreased (i.e., MF $j$ becomes fuzzier), the following lower limit satisfies the $\gamma$-condition:

$$b_{\gamma,j} = \frac{\ln(\kappa(\gamma))}{2\ln(d_{\gamma,j}/a_j)}.$$  

Combining the constraints yields to

$$\max(b_{\text{low}}, b_{\gamma,j}) \leq b_j,$$

if $b_j \geq b_{\text{high}}$

$$\max(b_{\text{low}}, b_{\gamma,j}) \leq b_j \leq \min(b_{\text{high}}, b_{\alpha,j}, b_{\beta,j}),$$

if $b_j < b_{\text{high}}$.

3) Dynamic Constraints for Parameter $c$: If $c_j$ is increased (MF $j$ is moving toward right), the following upper limit guarantees the fulfillment of $\alpha$-condition (only the right $\alpha$-condition needs to be taken into account):

$$c_{\alpha,j}^+ = \begin{cases} c_{\text{high}}, & \text{if } j = M_A \\ c_j + (I_L(\alpha, p_{j+1}) - I_R(\alpha, p_j)), & \text{if } j < M_A. \end{cases}$$

Furthermore, the following upper limit guarantees the fulfillment of $\beta$-condition (only the left $\beta$-condition needs to be taken into account):

$$c_{\beta,j}^+ = \begin{cases} c_j + (I_R(\beta, p_{j+1}) - I_L(\beta, p_j)), & \text{if } j > 1 \\ c_j + (\chi_{\text{low}} - I_L(\beta, p_j)), & \text{if } j = 1. \end{cases}$$

Finally

$$c_{\gamma,j}^+ = \begin{cases} c_{\text{high}}, & \text{if } j = M_A \\ c_j + \min(c_{j+1} - I_R(\gamma, p_j), I_L(\gamma, p_{j+1}) - c_j), & \text{if } j < M_A. \end{cases}$$

is the upper limit guaranteeing the fulfillment of $\gamma$-condition (only the right $\gamma$-condition needs to be taken into account).

If the value of $c$ is decreased (MF $j$ is moving toward left), the applied constraints are

$$c_{\alpha,j}^- = \begin{cases} c_j - (I_L(\alpha, p_j) - I_R(\alpha, p_{j-1})), & \text{if } j > 1 \\ c_{\text{low}}, & \text{if } j = 1. \end{cases}$$

$$c_{\beta,j}^- = \begin{cases} c_j - (I_R(\beta, p_j) - \chi_{\text{high}}), & \text{if } j = M_A \\ c_j - (I_R(\beta, p_j) - I_L(\beta, p_{j+1})), & \text{if } j < M_A. \end{cases}$$

$$c_{\gamma,j}^- = \begin{cases} c_{\text{low}}, & \text{if } j > 1 \\ c_j - \min(I_L(\gamma, p_j) - c_{j-1}, c_j - I_R(\gamma, p_{j-1})), & \text{if } j = 1. \end{cases}$$

Combining the constraints yields to

$$\max(c_{\alpha,j}^-, c_{\beta,j}^-, c_{\gamma,j}^-) \leq c_j \leq \min(c_{\alpha,j}^+, c_{\beta,j}^+, c_{\gamma,j}^+).$$

1Recall that $b_j$ can be increased only if $b_j < b_{\text{high}}$.

D. Genetic Operators

Five mutation and crossover operators are used. Some of them are not always applicable; therefore, when mutation or crossover is applied, one of the currently applicable operators is randomly selected by uniform chance. Crossover is applied with probability $P_c = 0.1 + (G/G_{\text{Tot}})$, where $G$ is the current generation, and $G_{\text{Tot}}$ is the total number of generations. If crossover was applied, mutation is applied with probability $P_m = 0.1$, and if crossover was not applied, mutation is always applied. This strategy is similar to strategy applied in [3].

Upper and lower limits for each modified parameter are computed according to Sections V-B and C and denoted by $L_{\text{upper}}$ and $L_{\text{lower}}$. Number of currently active MFs in an input variable is denoted by $M_A$ and a random real number by $r \in [0,1]$.

1) Mutation Operators: Operator 1 modifies the parameters of input MFs. First, the number of input variables that have at least two active MFs is determined. This number is denoted here by $n_{\text{active}}$. Then, out of $n_{\text{active}}$ input variables, $n_{\text{select}}$ of them are randomly selected, where $n_{\text{select}} \in [1, n_{\text{active}}]$ is a random integer. From each of these $n_{\text{select}}$ input variables, an active MF is randomly selected. Then, for each of them, a gbell parameter ($a$, $b$, or $c$) is randomly selected. They are denoted by $p_{i,l}$, where $i$ is 1, 2, or 3 depending upon which gbell parameter is modified, and $l$ is the index of an active MF in $P$ [see (17) and (18)]. Each $p_{i,l}$ is replaced by randomly selecting one of the following replacement formulas: $p_{i,l} \rightarrow p_{i,l} + r(L_{\text{upper}} - p_{i,l})$ or $p_{i,l} \rightarrow p_{i,l} - r(p_{i,l} - L_{\text{lower}})$.

Operator 2: The mutation operator 1 modifies input MF parameters individually; however, sometimes more drastic modification may be necessary. Therefore, this operator selects an input variable for which $M_A \geq 2$ and creates a new unevenly distributed partition with $M_A$ MFs using the algorithm defined in Section III-C.

Operator 3 modifies the rule base by randomly selecting $n_{\text{rulecond}}$ rule conditions $a_{i,j}$ [see (16)], where $n_{\text{rulecond}} \in [1, 10]$ is a random integer. The selected rule conditions are replaced with random rule conditions; however, as it is easier to obtain compact than accurate FMs [8], this operator favors nonzero-replacement conditions during the first half of the total number of generations $G_{\text{Tot}}$. Therefore, if $G < G_{\text{Tot}}/2$, then the probability that a replacement condition is selected from $[0, M_{\text{max}}]$ is $P_r = 2G/G_{\text{Tot}}$, and the probability that it is selected from $[1, M_{\text{max}}]$ is $1 - P_r$. When $G \geq G_{\text{Tot}}/2$, replacement conditions are always selected from $[0, M_{\text{max}}]$.

The resulting input fuzzy partition may not be transparent if some MFs have become active or inactive, thus resulting into highly overlapping MFs or gaps in the fuzzy partition. Thus, the set of these input variables that use different MFs in the rules before this operator is determined. Then, $M_A$ for each of these input variables is determined. For these input variables for which $M_A \geq 2$, new unevenly distributed partition with $M_A$ MFs is created. If $M_A < 2$, all nonzero conditions, if any, of that input variable are forced to zero. This operation is called repair operator, and it guarantees transparency of input fuzzy partition.
Operator 4 modifies a consequent \( s_i \), where \( i = 1, \ldots, R \), of a randomly selected active rule by replacing it by random consequent chosen from \([1, M_{\text{out}}]\). A rule is active if it has at least one nonzero-rule condition.

Operator 5 modifies the lateral displacement of a randomly selected active-output-MF center (an output MF is active if it is used in at least one of the active rules). The selected output-MF center \( o\), where \( i = 1, \ldots, M_{\text{out}} \), is replaced by randomly selecting one of the following formulas: \( o_i \leftarrow o_i + r(L_{\text{upper}} - o_i) \) or \( o_i \leftarrow o_i - r(o_i - L_{\text{lower}}) \).

2) Crossover Operators: All five crossover operators randomly select two FMs as parents and produce two FMs as children. They replace their parents in the offspring population. The crossover operators 1, 4, and 5 resemble the mutation operators 1, 4, and 5.

Operator 1 modifies the parameters of active input MFs using BLX-0.5 crossover [23], [32]. It can be applied to input variables, which have the same amount (at least 2) of active MFs in both parents. The number of input variables meeting these requirements is denoted by \( n_{\text{active}} \). Out of them, \( n_{\text{select}} \) are randomly selected, where \( n_{\text{select}} \in [1, n_{\text{active}}] \) is a random integer. For each of these \( n_{\text{select}} \) input variables, an active MF \( j \in [1, M_{\text{A}}] \) is randomly selected (the same \( j \) from both parents). Then, from each of these selected active MFs, a gbell parameter \((a, b, o)\) is randomly selected (the same parameter from both parents).

Let \( p_{i,l}^{k,1} \) and \( p_{i,l}^{k,2} \) denote the selected parameters from parents 1 and 2, respectively. The index \( i \) is 1, 2, or 3, depending on which gbell parameter is selected [see (17)]. The indexes \( l_1 \) and \( l_2 \) are determined according to (18). The parameters are replaced by randomly selecting either \( p_{i,l}^{k,1} \leftarrow p_{i,l}^{k,1} + r(\min(I, L_{\text{upper}} - p_{i,l}^{k,1})) \) or \( p_{i,l}^{k,2} \leftarrow p_{i,l}^{k,2} - r(\min(I, p_{i,l}^{k,2} - L_{\text{lower}})) \), where \( k = 1 \) and 2, and \( I = 0.5|p_{i,l_1}^{k,1} - p_{i,l_2}^{k,2}| \).

Operator 2: First, an input variable, for which at least one of the parents has at least two active MFs, is randomly selected. After this, all rule conditions and input MF parameters of this input variable are pairwisely swapped. Therefore, child 1 receives all the parameters of parent 1, except rule conditions and input MF parameters of the selected input variable, which are received from parent 2. Likewise, child 2 gets all the parameters of parent 2, except rule conditions and input MF parameters of the selected input variable, which are received from parent 1.

Operator 3 swaps some rules of the parents. It is applicable to those rules that are active in at least one of the parents. Out of these rules, \( N_{\text{select}} \) of them are selected and their rule conditions are pairwisely swapped \((N_{\text{select}} \text{ is a random integer chosen from } [1, 5])\).

After this operator, input fuzzy partitions may not be transparent. Therefore, for both children separately, the same repair operator as with the mutation operator 3 is applied.

Operator 4 modifies the rule consequents \( s_i \), where \( i = 1, \ldots, R \). This operator is possible for those rules that are active in at least one of the parents. The operator selects one of these rules randomly and swaps consequents of this rule.

Operator 5 modifies the lateral displacement of output MF centers. This operator is possible for those output MF centers that are active in both of the parents. Out of them, one is randomly selected from both parents (the same from both parents). They are denoted here by \( \alpha^1 \) and \( \alpha^2 \), where \( i = 1, \ldots, M_{\text{out}} \). They are replaced by randomly selecting one of the following formulas: \( \alpha_i \leftarrow \alpha_i + r(\min(I, L_{\text{upper}} - \alpha_i)) \) or \( \alpha_i \leftarrow \alpha_i - r(\min(I, \alpha_i - L_{\text{lower}})) \), where \( k = 1 \) and 2, and \( I = 0.5|\alpha_i^1 - \alpha_i^2| \).

VI. EXPERIMENTS

Our multiobjective GFS is validated using nine datasets, which represent different number of input variables and data points (see Table II). For all datasets, five-fold cross-validation was repeated six times \((6 \times \text{SCV})\) with different random seeds. The data partitions for Ele1, Ele2, Abalone, Mortgage, Treasury, and Computer problems were downloaded from KEEL Website.\(^2\) MG and Lorenz datasets were generated according to [3] and [20]. Finally, Gas dataset was obtained from the Website of Greg Reinse.\(^3\) For Mackey–Glass (MG), Lorenz, and Gas problems, the same data partitions as in the comparative study [20] were used. C4.5 was run with its default parameters defined in [29]. Population size was fixed to 100 and the number of generations was altered, such that, the same amount of fitness evaluations was used as in the comparative studies. The settings \( \alpha = 0.8, \beta = 0.05, \text{ and } \gamma = 0.25 \) are used in the experiments performed in Section VI-B-F. Furthermore, in VI-G, experiments with \( \alpha = 0.6, \beta = 0.4, \text{ and } \gamma = 0.1 \) will be performed in order to study the tradeof between transparency of fuzzy partitions and accuracy.

For six of the datasets (Ele1, Ele2, MG, Lorenz, Abalone, and Gas), there exist results of one or more recent GFSs presented in Table III. For these problems, the number of input and output MFs \((M_{\text{in}} \text{ and } M_{\text{out}})\) were selected the same as in the comparative studies. For treasury, mortgage, and computer problems, our method is compared against a baseline method. For these higher dimensional problems, \(M_{\text{in}}\) and \(M_{\text{out}}\) were both set to 3 in order to reduce the search space.

Since MOEAs are applied, it is interesting to visualize the Pareto fronts. However, it is not meaningful to visualize the Pareto fronts of all 30 CV runs for each dataset. The averaged results of the ith most accurate FMs from each of the 30 Pareto fronts were shown in [8] for five of the most accurate FMs (i.e., \( i = 1, \ldots, 5 \)). These averages were computed,

\(^2\)http://sci2s.ugr.es/keel/datasets.php
\(^3\)http://www.stat.wisc.edu/~reinsel/bjr-data/index.html
such that, none of the 30 Pareto fronts were excluded from computing the averages. Thus, in each of the Pareto fronts, there were at least five distinct FMs. In this paper, the maximum value of \( i \) (\( i_{\text{max}} \)) is not the same for all datasets, but depends on the minimum number of distinct FMs on the 30 Pareto fronts. More formally, \( i_{\text{max}} = \min(L_1, L_2, \ldots, L_{30}) \), where \( L_j \), with \( j = 1, \ldots, 30 \), is the number of distinct FMs on the \( j \)th Pareto front of a given dataset. Thus, the length of the averaged Pareto front equals to the length of the shortest Pareto front of the 30 runs.

Besides the Pareto fronts, the number of rules \( R \), rule conditions \( R_{\text{cond}} \), input MFs, and the number of input variables \( F \) for some of the \( i \)th most accurate FMs are tabulated. Moreover, the unequal variance \( t \)-test\(^4\) (denoted by \( t \)) with 95% confidence is reported for the MSE\(_{\text{trn}}\) and MSE\(_{\text{tst}}\). The same notations as in [6], [8], and [23] are used; \(*\) stands for the best averaged result in the column, + means that the performance of the corresponding row is worse than the best result, and \( \approx \) means that there is no significant statistical difference compared to the best result.

### A. Comparative Genetic Fuzzy Systems

The comparative approaches global lateral tuning (GL), global lateral tuning with rule selection (GL+S), global lateral amplitude tuning (GLA), and global lateral amplitude tuning with rule selection (GLA+S) minimize only one objective, namely, MSE, whereas the rest minimize two or more objectives simultaneously and obtain a set of Pareto optimal FMs. All GFSs use globally defined MFs. The approaches [6], [8], [23] create the initial populations using WM algorithm, whereas in [20], C4.5 algorithm is used. In this paper, the initial population is created by a method combining the benefits of C4.5 and WM algorithms.

Performance of GFS designs depends on their individual components, such as initialization method and MFs tuning strategy. For example, by applying different initialization methods, performance of a GFS can be significantly improved or deteriorated.

This is because appropriate initialization eases the derivation of better FMs due to reduction in the search space [15]. Because this paper and the comparative studies apply different initialization methods, the purpose of the results comparisons is not to assess the superiority of any individual components, but to assess the superiority of different approaches as a whole. Assessing the superiority of individual components is, of course, important, but requires another study in the future. It should be noted, however, that the results comparisons can be considered fair, because the same amount of fitness evaluations, the same data partitions, and the same amount of input and output MFs are used, as in the comparative studies. Also, our approach does not require any more \textit{a priori} knowledge about the datasets than the comparative methods.

To evaluate the transparency of fuzzy partitions, we follow [6], which states that two-tuple representation leads to more transparent fuzzy partition than three-tuple representation. Moreover, three-tuple representation is more transparent than classic three-parameter representation with static constraints. In [8] and [23], static constraints were defined, such that, MF parameters can vary within small intervals, whereas in [20], larger intervals were used. Therefore, we consider the transparency of fuzzy partitions in [20], which is the poorest among the comparative GFSs. Both two-tuple presentation and the proposed dynamic constraint approach maintain the transparency of fuzzy partitions at a good level. Since the approaches are quite different and because transparency of fuzzy partitions is a subjective matter, it is difficult to judge which one of them yields into more transparent fuzzy partitions. Therefore, their interpretability is considered equal.

### B. Estimating the Length of Low-Voltage Lines (Ele1)

For this problem, 50,000 fitness evaluations were used in this paper and in [6]. Table IV shows that GLA+S has the lowest MSE\(_{\text{tst}}\), and our most accurate FM (Final-1) has practically the same value. There is no statistical difference between the lowest MSE\(_{\text{tst}}\) and three of our most accurate FMs (Final-1, Final-2, and Final-3). The lowest MSE\(_{\text{tst}}\) is obtained by GLA, but again, there is no statistical difference between the lowest MSE\(_{\text{tst}}\) and three of our most accurate FMs. There is no clear

\[ \text{Table III}

\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
Ref. (Year) & Name & MFS tuning & Rule selection & Rule learning & Input variable selection: Initialization & Transparency of fuzzy partition \\
\hline
& GL+S & 2-tuple & No & Yes & No & Best \\
& GLA & 3-tuple & Yes & No & No & Good \\
& GLA+S & 3-tuple & Yes & No & No & Good \\
\hline
[8], [23] (2007-9) & TS-NSGA-II & 3-parameter + SSI & Yes & No & No & Average \\
& TS-SPEA2 & & & & & \\
& TS-NSGA-IIa & & & & & \\
& TS-NSGA-IIb & & & & & \\
& TS-SPEA2 & & & & & \\
& TS-SPEA2 & & & & & \\
\hline
[20] (2009) & C4.5+NSGA-II & 3-parameter + SLI & No & Yes & Yes & Poor \\
& This paper & Dynamic & No & Yes & Yes & Best \\
\hline
\end{tabular}

SSI and SLI stand for static constraints with small- and large-variation intervals, respectively.

\(^4\)Also called Welch’s \( t \)-test [33], [34]. If our multiobjective GFS could be compared to other GFSs in all problems, nonparametric tests would be preferred.
difference between different approaches for this problem, because the search space is small due to small amount of input variables. It is also noticed that although $M_{in}$ was set to 5, the initial FM uses on average nine input MFs. Therefore, one of the input variables is usually partitioned with four and the other one with five input MFs.

### C. Estimating the Maintenance Costs of Medium-Voltage Lines (Ele2)

This problem is more interesting as it contains four input variables. First, our multiobjective GFS was run for 50,000 fitness evaluations and compared to [6] and [23], which use the same amount of fitness evaluations. Table V shows that our multiobjective GFS has the lowest MSE in train and test sets. There is also statistical difference between our approach and all other approaches when MSE$_{tst}$ is considered. When MSE$_{trn}$ is considered, there is statistical difference between our approach and all other approaches, except TS-SPEA2Acc. Our FMs can also be considered as the most interpretable, because they are clearly the most compact, and the transparency of fuzzy partitions is at least the same as the comparative FMs (see also Table III).

Our approach was also run for 100,000 fitness evaluations (the same amount as in [8]). Table V shows that our FMs are the most accurate according to $t$-test. They are also clearly more compact than the FMs in [8]. Finally, because in [8], three-parameter MFs tuning with static constraints was used, the fuzzy partitions of our FMs can be considered more transparent.

### D. Predicting the Age of Abalone

This problem has eight input variables and a very high noise level. According to [8], usually the learning methods yields into similar accuracy. Thus, it may not be possible to improve the accuracy, but only to improve the interpretability, compared to existing methods in the literature. In this paper and also in the comparative study [8], the number of fitness evaluations was set to 100,000. According to Table VI, there is no clear difference in accuracy between different GFSs. The lowest MSE$_{trn}$ was obtained by TS-SPEA2Acc and the lowest MSE$_{tst}$ by our approach (Final-1). On the other hand, our approach presents a significant improvement in interpretability. Our FMs are clearly more compact than the comparative FMs. They have much less rule conditions and use much less input variables. Furthermore, according to Table III, our fuzzy partitions can be considered more transparent than the fuzzy partitions in [8].

---

**TABLE IV**

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Method</th>
<th>$R$</th>
<th>$R_{cont}$</th>
<th>MFs</th>
<th>$F$</th>
<th>SE$_{trn}$</th>
<th>SE$_{tst}$</th>
<th>t</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>[6]</td>
<td>GL</td>
<td>12.4</td>
<td>24.8</td>
<td>N/A</td>
<td>2.0</td>
<td>166674</td>
<td>11480</td>
<td>+</td>
<td>189216</td>
</tr>
<tr>
<td>GL+S</td>
<td>9.0</td>
<td>18</td>
<td>N/A</td>
<td>2.0</td>
<td>200081</td>
<td>7316</td>
<td>+</td>
<td>189844</td>
<td>22448</td>
</tr>
<tr>
<td>GLA</td>
<td>12.4</td>
<td>24.8</td>
<td>N/A</td>
<td>2.0</td>
<td>157604</td>
<td>9158</td>
<td>=</td>
<td>185810</td>
<td>18812</td>
</tr>
<tr>
<td>GLA+S</td>
<td>10.2</td>
<td>20.4</td>
<td>N/A</td>
<td>2.0</td>
<td>155404</td>
<td>9264</td>
<td>*</td>
<td>189472</td>
<td>20393</td>
</tr>
</tbody>
</table>

| Initial| 14.4 | 28.2| 9.0      | 2.0| 272050 | 17861     | + | 310160 | 45892 |
| This paper | 12.3 | 19.8| 8.9      | 2.0| 155785 | 9356      | = | 192319 | 31315 |
| Final  | 11.5  | 17.8| 8.9      | 2.0| 156911 | 10064     | + | 192813 | 31125 |
| Final  | 11.1  | 16.2| 8.7      | 2.0| 160178 | 12273     | = | 195730 | 40453 |

---

**TABLE V**

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Method</th>
<th>$R$</th>
<th>$R_{cont}$</th>
<th>MFs</th>
<th>$F$</th>
<th>SE$_{trn}$</th>
<th>SE$_{tst}$</th>
<th>t</th>
<th>t</th>
</tr>
</thead>
</table>

| [6]      | GL     | 65  | 260       | N/A | 4.0 | 23064     | 1479      | + | 25654  |
| GLA     | 65     | 260 | N/A       | 4.0 | 17950 | 1889     | + | 21212  |
| GLA+S   | 49.4   | 197.6| N/A      | 4.0 | 17538 | 2391     | + | 21491  |
| GL+S    | 49.1   | 196.4| N/A      | 4.0 | 18881 | 2669     | + | 22586  |

| [23]     | TS-NSGA-II | 48.1 | 192.4 | N/A | 4.0 | 16521     | 1636     | + | 20423  |
| TS-NSGA-IV | 41   | 164  | N/A   | 4.0 | 14488 | 965      | + | 18419  |
| TS-SPEA2Acc | 34.5 | 138  | N/A   | 4.0 | 11081 | 1186     | = | 14161  |
| TS-SPEA2  | 33     | 132 | N/A    | 4.0 | 13272 | 1265     | + | 17533  |

| Initial  | 26.2  | 61.6| 15.4    | 3.4 | 74719 | 15065    | + | 74727  |
| Final-1  | 25.0  | 51.5| 14.3    | 3.1 | 10861 | 1436     | = | 12336  |
| Final-2  | 24.8  | 49.7| 14.2    | 3.0 | 11314 | 1490     | = | 12731  |
| Final-3  | 24.5  | 47.5| 14.2    | 3.0 | 11942 | 1602     | + | 13551  |
| Final-4  | 24.1  | 45.6| 14.2    | 3.0 | 12742 | 2016     | + | 14080  |
| Final-5  | 23.9  | 44.0| 13.8    | 3.0 | 13571 | 2362     | + | 15100  |

<table>
<thead>
<tr>
<th>100000 fitness evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final-1</td>
</tr>
<tr>
<td>Final-2</td>
</tr>
<tr>
<td>Final-3</td>
</tr>
<tr>
<td>Final-4</td>
</tr>
<tr>
<td>Final-5</td>
</tr>
</tbody>
</table>

| [8]      | TS-SPEA2Acc-1 | 29.8 | 119.2 | N/A | 4.0 | 10325     | 1121     | + | 13935  |
| TS-SPEA2Acc-2 | 28.3 | 113.2 | N/A | 4.0 | 10496     | 1126     | + | 14268  |
| TS-SPEA2Acc-3 | 27.0 | 108.0 | N/A | 4.0 | 10835     | 1191     | = | 14460  |
| TS-SPEA2Acc-4 | 25.9 | 103.6 | N/A | 4.0 | 11217     | 1307     | = | 14806  |
| TS-SPEA2Acc-5 | 24.9 | 99.6 | N/A | 4.0 | 12194     | 2078     | + | 13417  |

| This paper | Final-1  | 24.9  | 49.1| 14.3    | 3.1 | 9366  | 887      | * | 10429  |
| Final-2  | 24.3  | 46.5| 14.2    | 3.1 | 9619  | 957      | = | 10713  |
| Final-3  | 23.3  | 44.0| 14.0    | 3.0 | 10007 | 1117     | + | 11270  |
| Final-4  | 22.9  | 41.7| 13.6    | 3.0 | 10339 | 1199     | + | 11663  |
| Final-5  | 22.2  | 39.4| 13.4    | 2.9 | 10948 | 1412     | + | 12261  |
E. Mackey–Glass, Lorenz, and Gas Problems

Our multiobjective GFS is compared to our former multiobjective GFS [20], which was run for 210 000 fitness evaluations on these problems. The same amount of fitness evaluations is used here. Table VII shows that our most accurate FMs are significantly more accurate than the most accurate FMs of our former study. On the other hand, they also contain much more rules and rule conditions than FMs in [20].

The least accurate FMs of the averaged Pareto fronts for each problem are also presented and denoted by Final-8, Final-4, and Final-9. One can notice that they are still more accurate than the most accurate FMs in [20]. On the other hand, they are also more complex with regards to number of rules and rule conditions. The number of input variables and the number of MFs is approximately the same. Table III shows that the FMs in [20] have the worst transparency of fuzzy partitions and our FMs have the best.

F. Higher Dimensional Problems: Treasury, Mortgage, and Computer Activity

Our approach was run for 100 000 fitness evaluations on these problems. To the best of our knowledge, there are no results of other GFSs available for these problems. Nonetheless, it is important to include a baseline method in order to have an idea about the performance of our approach. Thus, *Genfis3*, a fuzzy-c-means (FCM) clustering-based method was used to identify Mamdani FMs. This method is part of MATLAB’s Fuzzy Logic Toolbox 2. All settings, besides the type of FM, were kept at their default values and 6 × 5CV with the same data partitions as with our multiobjective GFS was performed.

Table VIII shows that our FMs are significantly more accurate than the comparative FMs. Moreover, they have less input variables and input MFs than the comparative FMs. The comparative FMs usually have less rules, but more rule conditions, than our FMs. By visual inspection, it was noticed that the fuzzy partitions by Genfis3 often contain many highly overlapping MFs and the UoD may not be strongly covered.

G. Fuzzy Partition Transparency Versus Accuracy Tradeoff

The experiments in Section VI-B–VI-F were performed with \( \alpha = 0.8, \beta = 0.05, \) and \( \gamma = 0.25 \). If one requires higher transparency of fuzzy partitions, the settings \( \alpha = 0.6, \beta = 0.4, \) and \( \gamma = 0.1 \) could be used. The 6 × 5CV procedures for all nine problems were repeated with these settings. The averaged results of the most accurate FMs are shown in Table IX along with the best and the worst results from Tables IV–VIII. In Fig. 4, the averaged Pareto fronts for five of the studied problems are shown. It is seen from Table IX and Fig. 4 that by improving transparency of fuzzy partitions, accuracy is deteriorated, but remains at a reasonable level.

Transparency of fuzzy partitions is evaluated against a fuzzy partition, which has three desirable properties: 1) The membership values at the intersections of neighboring MFs are always

---

3 At the time of writing the final version of this paper, this statement no longer holds true. There are recently published results available for some [26] and all [22] of these problems. However, the experimental setup in those papers differ significantly from the experimental setup of this paper. Thus, our results are not compared to them.
TABLE VIII
RESULTS COMPARISON FOR HIGHER DIMENSIONAL PROBLEMS

<table>
<thead>
<tr>
<th>Problem</th>
<th>Method</th>
<th>$R$</th>
<th>$R_{cand}$</th>
<th>MFs</th>
<th>$F$</th>
<th>$\text{MSE}_{\text{Err}}$</th>
<th>$\sigma_{\text{Err}}$</th>
<th>$t$</th>
<th>$\text{MSE}_{\text{Stat}}$</th>
<th>$\sigma_{\text{Stat}}$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treasury</td>
<td>Geni63</td>
<td>8.8</td>
<td>132.0</td>
<td>132.0</td>
<td>15.0</td>
<td>1.440</td>
<td>0.300</td>
<td>+</td>
<td>1.928</td>
<td>1.465</td>
<td>+</td>
</tr>
<tr>
<td>$F = 15$</td>
<td>Initial</td>
<td>8.8</td>
<td>19.2</td>
<td>7.4</td>
<td>3.4</td>
<td>1.203</td>
<td>0.437</td>
<td>+</td>
<td>1.128</td>
<td>0.810</td>
<td>+</td>
</tr>
<tr>
<td>Final-1</td>
<td>7.2</td>
<td>10.2</td>
<td>4.6</td>
<td>2.0</td>
<td>0.059</td>
<td>0.024</td>
<td>*</td>
<td>0.093</td>
<td>0.082</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Mortgage</td>
<td>Geni63</td>
<td>8.8</td>
<td>132.0</td>
<td>132.0</td>
<td>15.0</td>
<td>0.924</td>
<td>0.112</td>
<td>+</td>
<td>1.312</td>
<td>1.007</td>
<td>+</td>
</tr>
<tr>
<td>$F = 15$</td>
<td>Initial</td>
<td>19.4</td>
<td>55.0</td>
<td>10.6</td>
<td>4.4</td>
<td>0.737</td>
<td>0.444</td>
<td>+</td>
<td>0.449</td>
<td>0.150</td>
<td>+</td>
</tr>
<tr>
<td>Final-1</td>
<td>16.5</td>
<td>31.4</td>
<td>8.6</td>
<td>3.4</td>
<td>0.023</td>
<td>0.009</td>
<td>*</td>
<td>0.039</td>
<td>0.027</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Computer</td>
<td>Geni63</td>
<td>3.2</td>
<td>67.2</td>
<td>67.2</td>
<td>21.0</td>
<td>127.31</td>
<td>5.40</td>
<td>+</td>
<td>127.50</td>
<td>3.72</td>
<td>+</td>
</tr>
<tr>
<td>$F = 21$</td>
<td>Initial</td>
<td>30.0</td>
<td>219.7</td>
<td>38.8</td>
<td>15.6</td>
<td>452.73</td>
<td>494.18</td>
<td>+</td>
<td>453.80</td>
<td>496.53</td>
<td>+</td>
</tr>
<tr>
<td>Final-1</td>
<td>25.4</td>
<td>82.4</td>
<td>17.5</td>
<td>7.9</td>
<td>4.71</td>
<td>0.52</td>
<td>*</td>
<td>4.86</td>
<td>0.46</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4. Averaged Pareto fronts over 30 CV runs for Ele2 with 50 000 and 100 000 fitness evaluations, Abalone, Gas, Mortgage, and Computer problems. TP stands for transparent fuzzy partitions obtained by $\alpha = 0.6$, $\beta = 0.4$, and $\gamma = 0.1$. HT stands for highly transparent fuzzy partitions obtained by $\alpha = 0.6$, $\beta = 0.4$, and $\gamma = 0.1$.

The best and the worst results in Tables IV–VIII exclude the results of initial MFs.

TABLE IX
AVERAGED RESULTS OF THE MOST ACCURATE FM USING $\alpha = 0.6$, $\beta = 0.4$, AND $\gamma = 0.1$
Table X compares the averaged fuzzy-partition quality indexes of the most accurate FMs and the average length of the Pareto fronts with different settings of $\alpha$, $\beta$, and $\gamma$.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\gamma$</th>
<th>Ele1</th>
<th>Ele2/20000</th>
<th>Ele2/200000</th>
<th>MG</th>
<th>Lorenz</th>
<th>Abalone</th>
<th>Gas</th>
<th>Mortgage</th>
<th>Treasury</th>
<th>Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.05</td>
<td>0.25</td>
<td>0.32</td>
<td>0.28</td>
<td>0.29</td>
<td>0.28</td>
<td>0.30</td>
<td>0.23</td>
<td>0.28</td>
<td>0.26</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4</td>
<td>0.1</td>
<td>0.09</td>
<td>0.09</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
<td>0.09</td>
<td>0.08</td>
<td>0.08</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>0.1</td>
<td>0.05</td>
<td>0.25</td>
<td>0.36</td>
<td>0.29</td>
<td>0.32</td>
<td>0.44</td>
<td>0.44</td>
<td>0.57</td>
<td>0.48</td>
<td>0.50</td>
<td>0.16</td>
<td>0.13</td>
</tr>
</tbody>
</table>

Fig. 5. Ele2 (50,000 fitness evaluations). Examples of the most accurate FMs of one run using the same data partition. (Left) $\alpha = 0.8$, $\beta = 0.05$, $\gamma = 0.25$, $\text{MSE}_{\text{trn}} = 13277$, $\text{MSE}_{\text{tst}} = 12884$, $Q_{\text{Int}} = 0.27$, $Q_{\text{Mid}} = 0.23$, and $Q_{\text{Ext}} = 0.00$. (Right) $\alpha = 0.6$, $\beta = 0.4$, $\gamma = 0.1$, $\text{MSE}_{\text{trn}} = 18272$, $\text{MSE}_{\text{tst}} = 19439$, $Q_{\text{Int}} = 0.09$, $Q_{\text{Mid}} = 0.10$, and $Q_{\text{Ext}} = 0.00$.

0.5; 2) in the center of an MF, all other MFs receive membership value 0; and 3) at the extreme points $\chi_{\text{low}}$ and $\chi_{\text{high}}$ of UoD, one MF receives membership value 1. Three quality indexes are therefore computed for each fuzzy partition: 1) $Q_{\text{Int}}$: the maximum absolute difference from the desired intersection membership value 0.5; 2) $Q_{\text{Mid}}$: the maximum membership value of an MF in the center of another MF; and 3) $Q_{\text{Ext}}$: the maximum absolute difference from the desired membership value 1 at the extreme points of UoD. For a strong fuzzy partition, $Q_{\text{Int}} = Q_{\text{Mid}} = Q_{\text{Ext}} = 0$. One must, however, note that even a strong fuzzy partition can be poorly transparent, for example, when some of the MFs are very close to each other. These quality indexes do not take into account this kind of transparency aspects.

Table X compares the averaged quality-index values of the most accurate FMs for different settings of $\alpha$, $\beta$, and $\gamma$. Moreover, the average number $N_D$ and standard deviation $\sigma_{N_D}$ of distinct FMs on a Pareto front are shown. It is clearly seen that with the settings $\alpha = 0.6$, $\beta = 0.4$, and $\gamma = 0.1$, more transparent fuzzy partitions are obtained (i.e., the quality-index values are lower). The average length of Pareto fronts is, however, not clearly affected by the settings, but depends on the characteristics of each problem. As the number of rule conditions is one of the two fitness objectives, the Pareto fronts tend to be longer if the number of rule conditions in initial FM is high (see Tables IV–VIII).

Figs. 5 and 6 show examples of the most accurate FMs for Ele2 and Mortgage problems with different settings of $\alpha$, $\beta$, $\gamma$, $\text{MSE}_{\text{trn}}$, and $\text{MSE}_{\text{tst}}$. The quality indexes and the average number of distinct FMs on Pareto fronts are also shown.
and $\gamma$. It is seen that the fuzzy partitions are more transparent when $\alpha = 0.6$, $\beta = 0.4$, and $\gamma = 0.1$. One may notice that our approach performs input-variable selection, rule learning, granularity learning, and MF-parameters tuning. For example, it can be seen that one of the input variables for Mortgage problem is partitioned with three MFs, whereas the others are partitioned with two MFs. Moreover, these example FMs for Mortgage problem use only three or four input variables, even though the problem has 15 input variables.

**VII. CONCLUSION**

A dynamically constrained multiobjective GFS to learn the granularities of fuzzy partitions, tuning the MFs, and learning the fuzzy rules was proposed. It uses dynamic constraints, which enable application of three-parameter MFs tuning to improve the accuracy without deteriorating the transparency of fuzzy partitions. A new initialization method was also proposed. It combines the benefits of WM and DT algorithms, and reduces the number of rules, rule conditions, and input variables, while preserving the transparency of fuzzy partitions. Being a heuristic and suboptimal method, its main purpose is not to obtain very accurate and compact initial FMs, rather, its main purpose is to reduce the search space and, therefore, to ease the further optimization.

Nine benchmark problems having 2 up to 21 input variables were studied, and our multiobjective GFS was tested against 11 recently proposed multiobjective and monoobjective GFSs on six of these nine problems. It was seen that our approach always results into at least comparable accuracy and interpretability with the comparative approaches. Moreover, on some benchmark problems, it clearly outperformed some of the comparative approaches. On the rest three datasets, which have up to 21 input variables, it was tested against a FCM clustering method. It was seen that our FMs are more accurate and interpretable than the FMs obtained by FCM.

Our approach is suitable for both lower and higher dimensional problems. Suitability to higher dimensional problems is aided by the initialization method, which usually reduces the number of input variables. Naturally, if none of the input variables can be removed in initialization phase, the search space will be larger. This poses a challenge to any GFS and requires further research. By our approach, fuzzy partitions with different levels of transparency can be obtained by different settings of $\alpha$, $\beta$, and $\gamma$. It was shown that there exists a clear tradeoff between transparency of fuzzy partitions and accuracy. Finally, in this paper, regression problems were considered. However, our approach can be made suitable for classification problems as well [35].

**REFERENCES**

Pulkkinen and Koivisto: Dynamically Constrained Multiobjective Genetic Fuzzy System for Regression Problems


Pietari Pulkkinen received the M.Sc. degree from Tampere University of Technology, Tampere, Finland, in 2006. He is currently with the Tampere University of Technology. His research interests include software-computing methods, especially multiobjective genetic fuzzy systems, and applying them to real-world problems. He has authored or coauthored five international journal articles and four international conference papers. He is a Reviewer for several international journals, such as the International Journal of Approximate Reasoning. Mr. Pulkkinen is a Reviewer of the IEEE TRANSACTIONS ON FUZZY SYSTEMS.

Hannu Koivisto received the M.Sc. degree in electrical engineering and the Doctor of Technology degree from Tampere University of Technology (TUT), Tampere, Finland, in 1978 and 1995, respectively. From 2002 to 2007, he was the Head of the Automation and Control Institute, TUT, where he has been a Professor since 1999, and a Professor with the Department of Automation Science and Engineering. His current research interests include a wide range of topics in applications of artificial intelligence, such as control systems, decision support systems, and expert systems.

© 2009 IEEE. Reprinted with permission.
A Multiobjective Genetic Fuzzy System for Obtaining Compact and Accurate Fuzzy Classifiers with Transparent Fuzzy Partitions

Pietari Pulkkinen
Tampere University of Technology
Department of Automation Science and Engineering
P.O. Box 692, 33101 Tampere, Finland
pietari.pulkkinen@tut.fi

Abstract

A multiobjective genetic fuzzy system for classification problems is presented. Its advantage is that it uses 3-parameter membership function tuning with dynamic constraints. Therefore, the accuracy is improved without deteriorating the transparency of fuzzy partitions. The initial population is created with a method, which reduces the search space by removing irrelevant input variables. Then, multiobjective genetic fuzzy system optimizes the accuracy and complexity of the fuzzy classifiers and results into Pareto optimal set of compact and accurate fuzzy classifiers, which have transparent fuzzy partitions. The approach is compared to another multiobjective genetic fuzzy system and the advantages of our approach are shown.

1. Introduction

The main advantage of fuzzy classifiers (FCs) compared to other model types is their interpretability, that is, the ability to explain the reasoning behind the outcome. Thus, before applying an FC in practice, it is possible to verify not only that an FC has a satisfactory accuracy on a given data set but also that the fuzzy rules are reasonable. Identifying interpretable and accurate FCs based on data is a complex problem due to its large search space [6]. Because accuracy and interpretability are contradicting objectives, Pareto-optimal set of FCs presenting different tradeoffs between the objectives have been sought using multiobjective Genetic Fuzzy Systems (GFS) [6, 10, 14].

Accuracy is often measured by misclassification rate and interpretability by number of rules and rule conditions. This criteria for interpretability is only sufficient when the membership functions (MFs) are not tuned (e.g. uniformly distributed MFs or MFs based on a priori knowledge are used). However, MFs are often tuned because commonly there is no a priori knowledge about the distribution of MFs and because tuning usually improves the accuracy. In that case interpretability of FCs is not sufficiently characterized by number of rules and rule conditions but the transparency of fuzzy partitions needs to be taken into account. If that is neglected, interpretability of FCs is often deteriorated because it may not be possible to assign meaningful linguistic values for linguistic variables even if the FCs are compact.

Merging of highly similar fuzzy sets [1,13] and adjusting MF parameters in case that it is covering another MF [15] are ways to improve the transparency of fuzzy partitions. Moreover, in [8] highly overlapping MFs were avoided and the universe of discourse (UoD) was strongly covered by requiring that the intersection points of neighboring MFs must be between user specified limits. In [9] the approach [8] was extended to reduce the effects of relaxed covering [15] and a transparency index was optimized by a multiobjective GFS. That approach was further extended in [11] by introducing dynamic constraints. By applying dynamic constraints instead of optimizing a transparency index, the number of fitness objectives is reduced by one which increases the selection pressure [7].

Here we apply a multiobjective GFS for tuning the parameters of MFs, learning the granularities of input variables, and learning the fuzzy rules. We modify the method [11], which was designed for regression problems and make it suitable for classification problems. Our approach identifies FCs presenting tradeoffs between compactness and accuracy. Moreover, transparency of fuzzy partitions in each FC is guaranteed by dynamic constraints. The search space is reduced by applying an initialization method which reduces the number of input variables.

This paper is organized as follows. In the next section, the initialization method is introduced. Then, in section 3 the multiobjective GFS is presented followed by its performance evaluation in section 4. Finally, conclusions are given is section 5.
2. Population Initialization

Population is created in two parts. First only one FC is created. Then, the rest FCs of the population are created, such that, they have the same chromosome presentation as the first FC. Therefore, the first FC determines the maximum number of input variables, maximum number of input MFs per input variable and maximum number of rules for the rest FCs of the population.

2.1 Creation of the First FC of the Population

First, a decision tree (DT) using C4.5 algorithm [12] is created and all input variables which are not part of the DT are removed. DT is then converted into an FC according to [1]. That, however, often leads to overlapping input MFs. Therefore, the obtained input MF parameters are neglected and only the number of input MFs $M_{\text{DT}}$, for each input variable $j$ is considered important in this paper.

Then, input variables are partitioned with $\min(M_{\text{DT}}, M_{\text{in}})$ evenly distributed gbell MFs, where $M_{\text{in}}$ is the user provided maximum number of input MFs per input variable (usually between 3 and 9).

The rule base is created using Wang-Mendel (WM) algorithm [16] with some modifications. The original WM algorithm can create excessive amount of rules and rule conditions for some problems with many data points and/or input variables. Here, a rule is generated, such that, each data point is first classified by the DT and only those input variables, which were used by it to classify the data point are used for matching and become conditions of the generated rule. The consequent of the rule is simply the class label of the data point. Another modification is that an FC can contain at most $R_{\text{max}} = 30$ rules. If the number of generated rules is more than $R_{\text{max}}$, then $R_{\text{max}}$ rules are randomly selected. If that causes some input MFs to be inactive, there is a gap in at least one of the input fuzzy partitions. To avoid that, some randomly selected rules are replaced by rules, which make all the inactive input MFs active [11].

2.2 Creation of Remaining FCs of the Population

If the number of generated rules is at most $R_{\text{max}}$, then the rule bases of remaining FCs of the initial population are created by modifying the rule base of the first FC according to [10, 11]. Otherwise, the rule bases are created by randomly selecting rules from the generated rule base [11].

After creating the rule bases, the input MFs are created based on each individual rule base by first checking the number of active input MFs. For input variables with at least 2 active input MFs, unevenly distributed fuzzy partitions are created. For input variables with only 1 or 0 active input MFs, the active rule conditions, if any, are forced to don’t care conditions. That is done because fuzzy partitions with only 1 MF are not considered transparent [11].

3. Multiobjective Genetic Fuzzy System

NSGA-II, [3] a popular multiobjective genetic algorithm is used to optimize the initial population. Two objectives are to be minimized: (1) number of rule conditions and (2) number of misclassifications. The number of misclassification is constrained, such that, each FC must have less or equal number of misclassifications than a major class classifier.

3.1 Coding of FCs for Multiobjective GFS Optimization

Classification problems with $n$ input variables are considered. However, the initialization method selects $n_s \leq n$ of them. Hence, the data set is $Z = \{X, y\}$, where $X$ is $D \times n_s$ input matrix and $y$ is $D \times 1$ output vector (class label). Fuzzy classification can be performed as follows:

$$\text{Rule} \ i: \quad \text{if } x_1 \text{ is } B_{i,1}, \ldots, \text{and } x_{n_s} \text{ is } B_{i,n_s} \text{ then } g_i$$

where $i = 1, \ldots, R$, $R$ is the number of rules, $B_{i,j}, j = 1, \ldots, n_s$ is an input fuzzy set, $g_i \in \{1, \ldots, C\}$ is the rule consequent and $C$ is the number of different classes in data set. The degree of rule fulfillment is calculated for each data point $x_k, k = 1, \ldots, D$ as:

$$\beta_i(x_k) = \prod_{j=1}^{n_s} B_{i,j}(x_{kj}).$$

The output of an FC is the rule consequent associated to the rule with highest rule fulfillment (i.e. Winner takes all strategy). If several rules with different rule consequents have the highest rule fulfillment, classification of $x_k$ is rejected. The rule conditions are presented with an integer-coded matrix $A$ specifying for each rule $i = 1, \ldots, R$, which MF is used for input variable $j = 1, \ldots, n_s$:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n_s} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n_s} \\
\vdots & \vdots & \ddots & \vdots \\ a_{R,1} & a_{R,2} & \cdots & a_{R,n_s} \end{bmatrix},$$

where $a_{i,j} \in \{0, 1, \ldots, M_{\text{max},j}\}$ and $M_{\text{max},j}$ is the maximum number MFs in input variable $j$. Don’t care conditions are indicated by $a_{i,j} = 0$. Moreover, input variable $j$ is removed, if $\forall i, a_{i,j} = 0$. Rule $i$ is removed, if $\forall j, a_{i,j} = 0$. Input MF parameters to which each $a_{i,j}$ is referring are contained in a real-coded matrix $P$:

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,\delta} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,\delta} \\
\vdots & \vdots & \ddots & \vdots \\ p_{p,1} & p_{p,2} & \cdots & p_{p,\delta} \end{bmatrix},$$
where \( \rho \) is the number of parameters used to define an MF. In this paper \( \rho = 3 \), because gbell MFs are used.

The maximum number of MFs in an FC is denoted by \( \delta = \sum_{i=1}^{n} M_{\text{max}} \).

For any \( a_{i,j} \neq 0 \), the corresponding input MF parameters are \( p_{1,i}, p_{2,i} \) and \( p_{3,i} \), where

\[
l = \begin{cases} a_{i,j}, & \text{if } j = 1 \\ a_{i,j} + \sum_{k=1}^{j-1} M_{k}, & \text{if } j > 1 \\ a_{i,j}, & \text{if } j = 0 \end{cases}
\]

The rule consequent is defined in an integer-coded vector \( s = [s_1, s_2, \ldots, s_R]^T \), where \( s_i \) \( \in \{1, \ldots, C\}, i = 1, \ldots, R \). The total number of parameters to be optimized by a multiobjective GFS is \( \theta = Rn_s + \rho \delta + R, \) that is, the sum of the cardinalities of \( A, P, s, \).

### 3.2 Dynamic MFs Tuning Strategy

MFs tuning usually improves the accuracy, but may deteriorate the transparency of fuzzy partitions unless appropriate tuning strategy is used. We use dynamically constrained 3-parameter MFs tuning strategy [11], which starts from a transparent fuzzy partition and modifies gbell MF parameters \( a, b, \) and \( c \). However, only one parameter is modified at a time, such that, after each modification the following transparency conditions are met:

1. **Symmetry condition:** All MFs are symmetrical. We use gbell MFs, which are symmetrical by definition.
2. **a-condition:** At any intersection point of two MFs, the membership value is at most \( a \).
3. **\( \gamma \)-condition:** At the center of each MF, no other MF receives membership value larger than \( \gamma \). For gbell MFs, center is the parameter \( c \).
4. **\( \beta \)-condition:** UoD is strongly covered, that is, at each point of UoD at least one MF has membership value at least \( \beta \).

Generally, \( 0 < \beta, \gamma < 0.5 \) and \( 0.5 < a < 1 \) can be selected.

In this paper \( \beta = 0.05, \gamma = 0.25 \) and \( a = 0.8 \).

Fig. 1 (a) shows an example of unevenly distributed fuzzy partition with 5 gbell MFs. It is seen that the partition is transparent and meet the above conditions. If that partition is modified by, for example, decreasing the gbell parameter \( a \) of MF 2, the lower limit of \( a \) is computed depending on the parameters of its neighboring MFs 1 and 3 [11]. Fig. 1 (b) shows the result when the minimum value for \( a \) is assigned. If \( a \) were to be decreased more, the \( \beta \)-condition would be violated. Similarly, Fig. 1 (c) shows the result when minimum value for \( b \) is assigned and Fig. 1 (d) when maximum value for \( c \) is assigned. It is seen that after each modification the transparency conditions are still met.

It should be noted that the minimum and maximum values for parameters \( a, b, \) and \( c \) are not static but depend on the neighboring MFs.

### 3.3 Genetic Operators

Crossover is applied with probability \( P_c = 0.1 + \frac{G}{G_{\text{Tot}}} \), where \( G \) is the current generation and \( G_{\text{Tot}} \) is the total number of generations. If crossover was applied, mutation is applied with probability \( P_m = 0.1 \) and if crossover was not applied mutation is always applied. This strategy is similar to strategy applied in [2]. Altogether four mutation and cross-over operators are used. However, all four are not always applicable. Thus, one of the currently applicable operators is randomly selected by uniform chance.

The upper and lower limits for input MF parameters are computed according to dynamic MFs tuning strategy in section 3.2 and denoted by \( L_{\text{upper}} \) and \( L_{\text{lower}} \). Number of currently active MFs in an input variable is denoted by \( M_A \) and a random real number by \( r \in [0, 1] \).

#### 3.3.1 Mutation Operators

**Operator 1** modifies the parameters of input MFs. First, the number of input variables which have at least 2 active MFs is determined and denoted here by \( n_{\text{active}} \). Then, \( n_{\text{select}} \) of them are randomly selected, where \( n_{\text{select}} \in [1, n_{\text{active}}] \) is a random integer. From each of those \( n_{\text{select}} \) input variables an active MF is randomly selected. Then for each of them, a gbell parameter \( (a, b, \) or \( c) \) is randomly selected. They are denoted by \( p_{i,t} \) where \( i \) is 1, 2, or 3 depending which gbell parameter is modified and \( l \) is the index of an active MF in \( P \) (see also eqs. (4) and (5)). Each \( p_{i,t} \) is replaced by randomly selecting one of the following replacement formulas: \( p_{i,t} \leftarrow p_{i,t} + r(L_{\text{upper}} - p_{i,t}) \) or \( p_{i,t} \leftarrow p_{i,t} - r(p_{i,t} - L_{\text{lower}}) \).

**Operator 2** selects an input variable for which \( M_A \geq 2 \)
and creates a new unevenly distributed partition with \( M_A \) MFs. The unevenly distributed partition is created according to [11] and an example of it was shown in Fig. 1 (a).

**Operator 3** modifies the rule base by randomly selecting \( n_{\text{rulecond}} \) rule conditions \( a_{i,j} \) (see eq. (3)), where \( n_{\text{rulecond}} \in [1, 10] \) is a random integer. They are replaced with random rule conditions, however, as it is easier to obtain compact than accurate FCs [4], non-zero replacement conditions are favored during the first half of the total number of generations \( G_{\text{Tot}} \). So if \( G < G_{\text{Tot}}/2 \), probability that a replacement condition is selected from \([0, M_{\text{max}}]\) is \( P_2 = \frac{2G}{G_{\text{Tot}}} \) and the probability that it is selected from 
\([1, M_{\text{max}}]\) is \( 1 - P_2 \). When \( G \geq G_{\text{Tot}}/2 \) replacement conditions are always selected from \([0, M_{\text{max}}]\).

If this operator causes some MFs to become active or inactive, the resulting fuzzy partition may have highly overlapping MFs or gaps. Thus, the set of those input variables which now use different MFs in the rules is determined. Then, \( M_A \) for each of those input variables is determined. For those input variables for which \( M_A \geq 2 \), new unevenly distributed partition with \( M_A \) MFs is created. If \( M_A < 2 \) all non-zero conditions, if any, of that input variable are forced to zero. This operation is called repair-operator and it guarantees transparency of input fuzzy partition.

**Operator 4** modifies a consequent \( s_{i,j} \), \( i = 1, \ldots, R \) of a randomly selected active rule by replacing it by random consequent chosen from \([1, C]\). A rule is active if it has at least one non-zero rule condition.

### 3.3.2 Crossover Operators

**Operator 1** modifies the parameters of active input MFs using BLX-0.5 crossover [5]. It is applicable to those input variables containing the same amount (at least 2) of active MFs in both parents. The number of input variables meeting those requirements is denoted here by \( n_{\text{active}} \). Out of them, \( n_{\text{select}} \) are randomly selected, where \( n_{\text{select}} \in [1, n_{\text{active}}] \) is a random integer. For each of those \( n_{\text{select}} \) input variables, an active MF \( j \in [1, M_A] \) is randomly selected (the same \( j \) from both parents). Then from each of those selected active MFs, a gbell parameter \((a, b, \text{or} c)\) is randomly selected (the same parameter from both parents).

Let \( p_{i_{	ext{l}} j_{	ext{l}}} \) and \( p_{i_{	ext{l}} j_{	ext{l}}} \) denote the selected parameters from parents 1 and 2, respectively. The index \( i \) is 1, 2, or 3 depending on which gbell parameter is selected (see eq. (4)). The indexes \( l_1 \) and \( l_2 \) are determined according to eq. (5). The parameters are replaced by randomly selecting either \( p_{i_{	ext{l}} j_{	ext{l}}} \leftarrow p_{i_{	ext{l}} j_{	ext{l}}} + r(\min(I, U_{\text{upper}} - p_{i_{	ext{l}} j_{	ext{l}}}))) \) or \( p_{i_{	ext{l}} j_{	ext{l}}} \leftarrow p_{i_{	ext{l}} j_{	ext{l}}} - r(\min(I, p_{i_{	ext{l}} j_{	ext{l}}} - L_{\text{lower}}))) \), where \( k = 1, 2 \) and \( I = 0.5p_{i_{	ext{l}} j_{	ext{l}}} \).

**Operator 2:** First, an input variable, for which both parents have at least two active MFs, is randomly selected. After that all rule conditions and input MF parameters of that input variable are pairwise swapped. So the child 1 receives all the parameters of parent 1, except rule conditions and input MF parameters of the selected input variable, which are received from parent 2. Likewise, child 2 gets all the parameters of parent 2, except rule conditions and input MF parameters of the selected input variable, which are received from parent 1.

**Operator 3** swaps some rules of the parents. It is applicable to those rules which are active in at least one of the parents. Out of those rules, \( N_{\text{select}} \) of them are selected and the rule conditions of those rules are pairwise swapped (\( N_{\text{select}} \in [1, 5] \) is a random integer).

After this operator, input fuzzy partitions may not be transparent. Thus, for both children separately the same repair-operator as with the mutation operator 3 is applied.

**Operator 4** modifies the rule consequents \( s_{i,j} \), \( i = 1, \ldots, R \). It can be applied to those rules which are active in at least one of the parents. It selects one of those rules randomly and swaps consequents of that rule.

### 4. Experiments

Two well-known problems, namely Glass and Wine, were studied to validate our proposal. Glass problem has 214 data points, 9 input variables, and 6 different classes. Whereas Wine problem has 178 data points, 13 input variables, and 3 different classes. 10-fold cross-validation was repeated 10 times (i.e., 10×10-CV) with different random seeds. So, for both problems our approach was run 100 times. The data partitions were downloaded from[1]. The maximum number of input MFs per input variable \( M_{\text{in}} \) was set to 7 for both problems. The settings for C4.5 were kept at their default values defined in [12].

Our results are compared to a recently published multi-objective GFS [10], which also used 10×10-CV to evaluate its performance. In [10], the number of fitness evaluations was set to 100000. We follow that and set the number of generations to 5000 and the population size to 200. The approach [10] does not use dynamic constraints to guarantee the transparency of fuzzy partitions. Therefore, it is expected that it should yield into better accuracy than our approach, since there is a tradeoff between transparency of fuzzy partitions and accuracy [11]. Therefore, if the accuracy in this paper is close to the accuracy in [10], it can be considered that our approach has good performance.

To compare the approaches, the averaged results of the FCs with the lowest training error rate \( E_{\text{trn}} \) over 100 runs are shown in Table 1 and denoted by Final. The averaged results of the first FC of the initial population (see section 2.1) are also reported and denoted by Initial. It is seen that the initial FCs on average use only 3.3 of the possible 13 input variables \( F \) for Wine problem and 8.0 of the possible 9

---

input variables for Glass problem. T-test (t) with 95% confidence is reported for the error rate on train and test sets ($E_{trn}$ and $E_{tst}$). The same notations as in [4] are used; * stands for the best averaged result in the column, + means that the performance of the corresponding row is worse than the best result, and = means that there is no significant statistical difference compared to the best result. Finally, the best results of each column for each problem are indicated with boldface.

### 4.1 Wine Problem

Table 1 shows that the comparative approach [10] achieves the lowest $E_{trn}$ and the proposed approach the lowest $E_{tst}$ for Wine problem. However, according to T-test, there is no statistical difference in accuracy. The FCs in [10] have slightly less rules $R$, rule conditions $R_{cond}$ and MFs, but our FCs have slightly less input variables $F$. Finally, the fuzzy partitions by our method are more transparent than the fuzzy partitions in [10]. One of them is shown in Fig. 2.

### 4.2 Glass Problem

According to Table 1 the most accurate FCs in [10] are statistically more accurate than the most accurate FCs in this paper when $E_{trn}$ is considered. Nonetheless, there is no statistical difference when $E_{tst}$ is considered, which shows that the proposed approach has good generalization capability. Moreover, our most accurate FCs are more compact and like Figs. 3 and 4 show, our fuzzy partitions are much more transparent.

The averaged Pareto fronts\(^2\) are shown in Fig. 5. It is seen that although our method has better transparency of fuzzy partitions, it is competitive in accuracy as well. Especially the testing accuracy is practically the same as in [10].

### 5. Conclusions

A multiobjective GFS for obtaining a Pareto optimal set of compact and accurate FCs with transparent fuzzy parti-
tions was presented. The approach is started with an initialization method which reduces the search space by removing irrelevant input variables. Then, a multiobjective GFS simultaneously learns rules, granularities of input variables, and MF parameters. It uses 3-parameter MFs tuning with dynamic constraints, which improves the accuracy and maintains the transparency of fuzzy partitions.

The proposed approach was compared to another multiobjective GFS and although the comparative approach does not maintain the transparency of fuzzy partitions, the accuracy of our FCs was competitive. Furthermore, transparency of fuzzy partitions obtained by our method was clearly better than in the comparative study.

References


