From Approximative to Descriptive Fuzzy Models

Javier G. Marín-Blázquez

Ph.D.

University of Edinburgh

2002

This thesis is dedicated to the loving memory of my father, Antonio.

Dad, now it is you who have all the answers.

Esta tesis está dedicada a la querida memoria de mi padre, Antonio.

Papá, ahora eres tú quien tiene todas las respuestas.

Abstract

AI knowledge based systems have proven a very valuable tool to understand and model complex real-world problems. Unfortunately, human experts are often required to provide the knowledge or rules. Due to limited availability and sometimes experts' reluctance, or even inability, to give such expertise, knowledge acquisition has become the bottle-neck in the development of knowledge based systems. However, there may well be a sensible amount of data that represents typical description of the domain problems. Therefore, data-driven techniques for automated generation of rules have been developed over the years to assist knowledge acquisition. While there have been efforts to make the automatically generated models transparent and readable, in most cases complexity ends up taking over.

Fuzzy sets offers a way to deal with vague, imprecise or inaccurate information while reducing the complexity of knowledge representation. However, most of fuzzy rule generation methods, while fast, accurate and easily scalable to high dimensional problems, follow the so-called *approximative approach*, which works by creating and tuning the fuzzy rule bases and the fuzzy sets to best fit the data. Opposing this stands the *descriptive approach*, where semantics are as important as accuracy and the definition of the fuzzy sets is human given, thereby representing human interpretable concepts. Unfortunately, the methods to obtain descriptive models are generally rather slow, inaccurate and poorly scalable. It would be desirable to create a method to use the benefits of fast and efficient generation of the approximative approaches with the clarity and comprehensibility of descriptive approaches. The purpose of this thesis is the development of such a method.

The thesis presents an effective and efficient approach for translating fuzzy rules that use approximative sets (accurate but unreadable) to rules that use descriptive sets and linguistic hedges of predefined meaning. It works by first generating rules that use approximative sets from training data, using a fast and accurate approximative algorithm that already exists. Then the resulting approximative rules are translated into descriptive ones. First, a heuristic conversion is performed to obtain a crude descriptive translation. Such a heuristically generated descriptive fuzzy model is then used to initialize a multi-objective GA. The GA, guided by the novel *functional equivalence* objectives, will fine-tune the heuristic translation into the final descriptive fuzzy rule set.

Hedges that are useful for supporting such translations are provided. This thesis presents an improved version of more effective hedges specifically devised for trapezoidal fuzzy sets, to be applied to dilate or concentrate a given set by expanding or shrinking its constituent parts. It also introduces three new hedges not existing in the literature.

Considerable experimental studies have been carried out, on the issues of the accuracy and transparency of the descriptive rules generated by the proposed approach. These include comparative analysis between alternative modelling approaches,

demonstrating the success of the present work. In particular, it is shown that the translated rules are functionally equivalent to the original approximative ones, or a close equivalent given search time restrictions, while reflecting their underlying preconceived meaning. Thus, fuzzy descriptive models can be obtained by taking advantage of any existing approach to approximative modelling which is generally efficient and accurate, whilst employing rules that are comprehensible to human users.

Acknowledgements

A PhD is much more than a nine to five job for a few years. A full-time PhD, at least in my case, takes over your whole life. It is for this reason that, in this section of the acknowledgements, there are so many people I wish to thank for the help, support and encouragement over all these years. Thank you all.

On the academic side there are three main figures of great influence, my three supervisors: **Qiang Shen**, **Antonio Gómez Skarmeta** and **Peter Ross**. I have been extremely fortunate in enjoying the privilege of being supervised by them. It may seem a cliché but without them this work would have never been completed. Their brilliant guidance, support and inspiration really made the difference. I will always being indebted to them. Thank you very, very much.

I wish to thank also my examiners, **John Hallam** and **Trevor Martin**. Their insightful corrections and suggestions have given a valuable touch of quality to the final version.

There were many PhD students, in the level E and F of the tragically lost, in the great fire of Edinburgh in 2002, South Bridge building that helped a lot in my research. They contributed from answering questions about LaTeX up to major issues in the research, no mention of the hundreds of coffees shared. I want to thank all of them and specially the fellow members of the AQR group.

This is also the place to thank all the funding bodies that provided the money to support me here in Edinburgh. In particular the **Caja de Ahorros del Mediterraneo (CAM)**, the **Seneca Foundation**, the **Marín-Blázquez Foundation** (my Mum) and myself (with hundreds of hours of tutorials, markings and lecturing). Their investment has been productive after all.

I have to thank Science in general as well. I am lucky to live and in an enlightened part of the world in an enlightened period of history, but it has not been always like this. Many scientists have had to fight, even risking their own lives, for the pursuing of knowledge and to lighten a world that is, sadly, still full of ignorance, superstitions and darkness. The story of Science, from the Babylonian star-gazers, through the Greek mathematicians and philosophers, up to the current colleagues, is full of the bravest, hardest working and most brilliant minds in human history. I want to thank them all for bringing light and knowledge to the world, for inspiring me, for teaching me and provide me with the pleasure of learning, and learning truth. Anything of use I may end up producing to the Science community will be thanks to all and each of them.

Many, many friends have supported me in all these years in Edinburgh, away from home and family. Most of them were, as myself, staying at Edinburgh for just a few years. They came and went but most of them left a durable print and in a way still linger. They have been indeed my family here and made me feel at home. The list is incomplete (I have a poor memory for names) but anyway most of them will never read a line of this thesis, not even the title. My thanks to Rubén, Gorka, Carolina, Charlie,

Silvia, Juan, Tomás, Flo, Sandra, Joaquín, Mark, Chris, Esti, Esther (Yaso), Laura, Yolanda, Inés, Susi, Martina and so many others. You made Edinburgh unforgettable.

Nevertheless among all of them some have been particularly dear and close to me and I wish to thank them specially: **Angel de Vicente**, **Kike Collantes**, **Juan Manuel Vazquez** and **Sonia Schulenburg**. They have been my brothers and sister here. You will always be with me.

There are also many friends in Spain that have supported me in the hard times, that always welcomed me back every single visit, that have proved time and again the so good friends they are. Many thanks to **Camino**, **Pepe**, **Cari**, **Javi**, **Ana**, **Mariano**, **Ginés** and **Humberto**. You kept my drive to go back to Spain. You made me know I will always be welcomed back.

I could have never done this without the full support of my real family. It was a sacrifice for all of them. They gave me the chance to fulfil my dreams and improve, beyond my own hopes, my education. My brothers **Antonio** and **Juan María**; and sister **Piedaita** shouldered my burden in the family's responsibilities, allowing me the freedom and time to do this PhD. I could not have done it without them, this is also their work. Thank you very much.

Finally I wish to thank very, very specially, my Mum, **Piedaita Marín-Blázquez**. There is no sea, no distance, no mountain able to prevent a Spanish mother's support. She has never put a foot in Edinburgh, but that did not stop her to help me enormously, and in an unconditional way. Muchas gracias mamá.

Declaration

I hereby declare that I composed this thesis entirely myself and that it describes my own research.

Javier G. Marín-Blázquez
Edinburgh
May 16, 2003

Table of Contents

List of Figures

Chapter 1

Introduction

> "Beware of the man who knows the answer before he understands the question."
>
> *Chinese Proverb*

This work is about artificial intelligence helping natural intelligent beings to understand real-world systems. There are many processes or systems where only data is available but there are no models that define the system behavior or describe the relations among the variables/factors involved. However, computers have been used as the tool to implement artificial intelligence ideas and techniques. Thanks to the power of computers to make calculations and compare numbers, predictive systems that work with numerical data have been developed over the years and have started to be useful for diverse human activities. Nevertheless, with the blessings of usefulness and accuracy usually come the curses of overcomplexity and obscureness.

Although there have been efforts to create transparent models, in most cases complexity ends up taking over. In particular, complexity has often been dealt with using obscure mathematical techniques, that are difficult to understand for the non-initiated. Moreover, it is very common that the data itself, gathered from the systems to be modeled, is either inaccurate, imprecise or difficult to express as a given quantity, making much of the information available vague and rather

unreliable. Fuzzy logic has proved successful to substantially alleviate the latter two problems. It offers a way to deal with vague, imprecise or inaccurate information while reducing the complexity of representation. Even lay-people with little mathematical background can easily grasp the concepts of fuzzy sets, making the models which are generated using fuzzy logic readable. This work uses fuzzy logic to create if-then rule style models to describe processes/systems where data is available using artificial intelligence techniques. Such rules use human words, defined by humans. Just by looking at the rules, humans can figure out how the system may behave. It is in this manner that this work is about *artificial intelligence helping natural intelligent beings to understand systems.*

1.1 Fuzzy Logic

Computing with words is a fundamental contribution of fuzzy logic [Zadeh 96]. This is feasible via the utilization of linguistic variables which are variables whose values can be words rather than numbers. These words can be interpreted as semantic labels to the fuzzy sets employed within the fuzzy models [Zadeh 75]. Thus, human comprehensible computer representation of the domain problems concerned can be created when desired.

In applications such as, for example, systems monitoring, medical diagnosis, etc. domain attributes often emerge from an elusive vagueness, a re-adjustment to context or an effect of human imprecision. The use of the soft boundaries of fuzzy sets, namely the graded memberships, allows subjective knowledge to be incorporated in describing these attributes and their relationships. Fuzzy techniques have proven to be very successful for creating, for example, robust controllers and user-friendly classifiers [Abe 98, Wang & Mendel 92, Roubos & Setnes 01] to address such problems. Even when precise knowledge is available, fuzziness may be a concomitant of complexity involved in the reasoning process. Among the interesting features of fuzzy approaches

is the potential of fuzzy production rules in attaching meaningful labels to the fuzzy sets [Zadeh 75], thereby allowing a human comprehensible representation of the system under consideration.

1.1.1 Approximative approach

Fuzzy rule bases are typically assumed to be given by domain experts. However, the acquisition of knowledge on rule structures often forms the bottleneck to advancing the success of fuzzy systems (and indeed of any knowledge-based systems) in practice [Rauma 96], though the linguistic labels or fuzzy sets that are used within the rules may be subjectively defined. For many applications, there exists a considerable volume of historical data obtained by observing the behavior of the system concerned. It is therefore desirable to be able to automatically generate rules from given data.

Many techniques exist for this, most of which follow the so-called *approximative approach* (or precise fuzzy modeling), which works by creating and tuning the fuzzy rule bases to best fit the data. The word *approximative* is used here instead of approximate to mirror the word *descriptive* in descriptive modeling which is itself an approximate approach. The rules generated are not encoded to keep the meaning of the linguistic labels of the fuzzy sets used. Such an approach is, under minor restrictions, functionally equivalent to neural networks [Jang *et al.* 93] and the resulting systems offer little explanatory power over their inferences. In fact, important automatic model generation techniques such as clustering, variations of hill climbing and genetic algorithms are all data-driven. When employed unrestrictedly for fuzzy modeling, they tend to create fuzzy sets that fit the data extremely well but that usually lack features which are considered important to make it easy for human users to interpret the resulting model and its reasoning [Valente deOliveira 99].

The most effective and common approximative fuzzy rule systems are usually generated by locating groupings (or clusters) of data points with similar output. Fuzzy sets are created as the convex closure of the projection of such points on each variable

axis. Finally rules are created as the logical conjunctions of the fuzzy sets so generated. Later, the fuzzy sets and the rules themselves are improved using any optimization technique to reduce modeling errors. The type of the fuzzy sets used is chosen to best suit the generation or tuning algorithm, with the piece-wise ones being the most common in earlier times and curves or ellipsoidals in later works.

With the development of neuro-fuzzy systems [Jang *et al.* 97], radial basis functions and other s-curves have been more and more used to represent fuzzy membership functions as they can be continuously differentiated unlike linear piece-wise functions. Such a differentiable property is of utmost importance to apply a gradient descendant technique (like the back-propagation algorithm) that is the heart (and weakness due to local maxima stagnation) of neuro-fuzzy systems. Nevertheless, such curves are generated with parameters that are rather obscure for non-technical people (perhaps, with the center the only clear one), and only graphical plots of them help give an idea of the shape of the resulting membership functions.

It is, therefore, important to note that following such generation/tuning techniques little serious interpretative power can be claimed for approximative fuzzy rule systems.

1.1.2 Descriptive approach

Opposing approximative modeling stands the *descriptive approach* (or linguistic fuzzy modeling). From this approach point of view, semantics are as important as accuracy. The definition of the fuzzy sets is human given. For pure descriptive modeling no changes to such definition are allowed. An example of a descriptive fuzzy partition can be seen in figure 1.1. As humans can not process efficiently more than 7 ± 2 different entities in the short term memory [Miller *et al.* 60], a linguistic variable of practical use should not have more than such a number of possible different labels.

However, this constraint leads to coarse partitions of the underlying value ranges of the linguistic variables and hence affects the accuracy of the model to be built. The descriptive sets used induce a fuzzy grid in the product space of the domain variables.

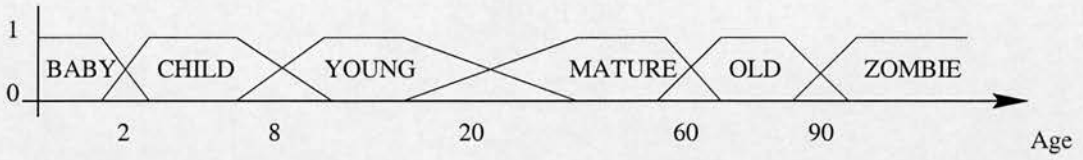


Figure 1.1: Example of descriptive fuzzy partition using trapezoidal fuzzy sets

As few or even no modifications are permitted, the grid, and the hyperboxes delimited by it, is almost fixed. Often these hyperboxes may contain examples of different output states and, as they are fixed, there is no way to separate these outputs directly. An example of this can be seen in figure 1.2. The dots reflect some data about age versus height. The blue box reflects the rule IF Age is YOUNG then Height is MEDIUM. As can be seen not all young people fall in the medium height set and no other rule with young in the antecedent can be generated (in a consistent rule base there can not be two rules with the same antecedent but different consequents, or else there would be no way to know which one to use). The only way to cover all the young data (in particular the red circled ones that can not belong to a set other than to young) would be to modify the set medium making it bigger so it covers all young examples. Yet doing so would change the meaning of medium.

One of the important disadvantages of descriptive modeling is that the best rules are usually generated by an exhaustive search in the given data space (e.g. [Lozowski *et al.* 96, Au & Chan 98]). This can only be done for problems involving a small number of variables/labels due to the potential combinatorial explosion. Even with a manageable size of variables/labels, the automatic generation of descriptive rules is generally either very slow, or very inaccurate, whilst many existing approximative methods are able to find very accurate rules rapidly. Thus, the question becomes if there exists a way to generate descriptive rules using a variation of these approximative methods.

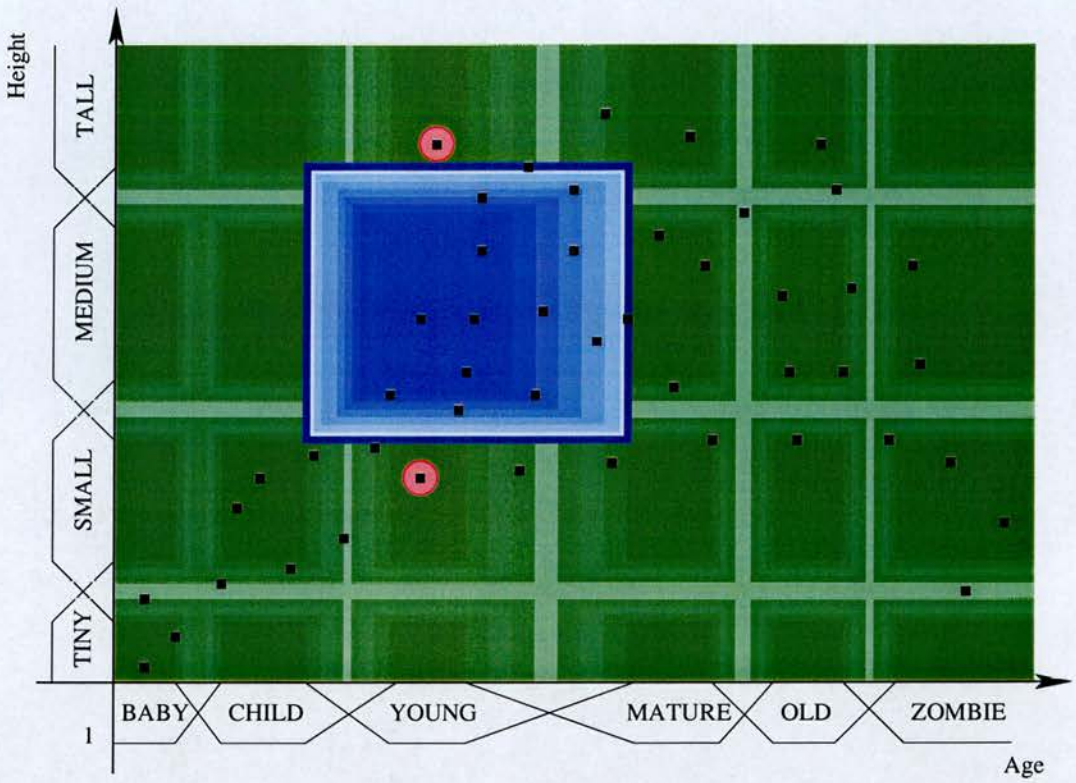


Figure 1.2: Example of descriptive fuzzy grid using trapezoidal fuzzy sets, and the descriptive rule IF Age is YOUNG then Height is MEDIUM

1.1.3 Linguistic Hedges

It is possible to implicitly modify fuzzy rule bases without disrupting the definition of the underlying fuzzy sets, by the use of linguistic hedges [Zadeh 75] which allow more freedom in manipulating the hyperboxes. A linguistic hedge modifies the shape of a fuzzy set's definition, causing changes in the membership function. That is, a hedge transforms one fuzzy set into another. The meaning of the transformed set can be extracted from that of the original set and that embedded in the hedge applied. Not all pure descriptive methods (which do not allow any redefinition of the fuzzy sets used) support the addition of hedges though, including the well established work such as that reported in [Wang & Mendel 92]. When no hedges can be applied an increase

in the number of fuzzy sets, the addition of confidence factors or prioritising some data as critical, may increase the performance. However, these methods typically also give rise to a loss in the interpretability.

1.1.4 Fuzzy rule systems tuning and pseudo-descriptive approaches

Approximative approaches avoid the problem of fixed hyperboxes by changing the definitions of the fuzzy sets and hence of the hyperboxes themselves. This ruins the underlying prespecified meaning attached to the fuzzy labels which have a natural appeal to the common-sense understanding of the words used. This is particularly so when a free or weakly constrained modification of fuzzy sets is carried out, even for those well-established approaches as reported in [Nauck & Kruse 98, Nauck *et al.* 97]. Until recently, literature paid little attention to this side-effect of using an approximative model and focused on the accuracy of derived models (some even regarding such models as descriptive or interpretable), or simply maintained a descriptive model and accepts high modeling errors.

Approximative tunings use almost any known optimization technique, from gradient descent (back-propagation) and all the family of hill climbers to genetic algorithms. Many of them change both parameters of the fuzzy sets and the structure of the rules themselves. Freedom is absolute in these systems so the only objective is to improve the modeling accuracy. Nevertheless, transparency and readability is so desirable that there have been several attempts to obtain it, keep it or regain it. The ones that do not impose the use of unmodified human defined sets can be seen as *pseudo-descriptive* approaches.

Among the recent attempts made to regain some of the approximative fuzzy systems' transparency are those as presented in [Setnes & Roubos 00, Setnes *et al.* 98b, Setnes *et al.* 98a]. This approach works by reducing otherwise possibly many antecedent approximative sets into a manageable number that possess

interesting properties [Valente deOliveira 99]. However, the linguistic labeling is done, when possible, *a posteriori*; the labels are attached to fuzzy sets generated by an approximative method but not to those given by humans. As the fuzzy sets used are self-clustered from training data and then given an artificial name, they may not have an intuitive interpretation. Human users of the resulting fuzzy systems have to do with the "friendly" words produced by the computer rather than the other way around.

Significant work has been proposed to obtain descriptive explanations of approximative models [Sugeno & Yasukawa 93], where each approximative rule fired is translated from one approximative hyperbox to one closest descriptive hyperbox. This represents an important departure from pure approximative modeling approaches. However, it adds on additional runtime cost and the explanation generated may not be sufficiently accurate due to the one-to-one approximate translation.

Note that, *pseudo-descriptive* approaches may also include approximative methods where certain restrictions are imposed to allow for some descriptive features [Nauck *et al.* 97]. Nevertheless, the fact that the fuzzy sets are modified or generated by computer implies a lack in obtaining a real descriptive model which employs such sets. It is for this reason that such an approach has been referred to as pseudo-descriptive.

Pure descriptive models do not allow much freedom for tuning, the only option for these systems is to change the structure of the rules and test different combination of antecedents. The number of combinations can be huge if many variables and many labels per variable are involved, making this process computationally intractable. Even so, in most cases the fixed hyperboxes do not cover the data in a convenient way, and the only solution seems to be changing the definition of the sets.

Thus, it is desirable to modify fuzzy models without disrupting the definition of the underlying fuzzy sets. As mentioned above, this is possible by the use of linguistic hedges, which allow fine modification of the human defined fuzzy sets without

changing the original definition, and hence, they offer more freedom in manipulating the hyperboxes.

1.2 Major achievements

This thesis presents an alternative approach, based on the initial investigations as reported in [Marín-Blázquez *et al.* 00, Marín-Blázquez & Shen 01], for producing descriptive fuzzy systems with a two-step mechanism. The first is to use an approximative method to create accurate rules and the second to convert the resulting approximative rules to descriptive ones. The conversion is, in general, one-to-many, implemented by a heuristic method that derives potentially useful translations and then by performing a fine tuning of these translations via evolutionary computation. Both steps are computationally efficient. The resultant descriptive system is ready to be directly applied for inference; no approximative rules are needed at runtime. Note that the work described here is focused on classification tasks. A schematic graph of the process can be seen in Figure 1.3.

The overall conversion process proposed is guided by *functional equivalence* rather than by similarity between approximative fuzzy sets and predefined descriptive ones in the antecedent part of the rules. The ultimate objective is to obtain a whole descriptive ruleset and to use it to perform the inference direct, thereby providing not only human comprehensible models but also straightforward explanation of the reasoning based upon the resulting models. To allow more flexible modeling results, such that predefined fuzzy sets may be modified in their effects within the resulting descriptive rules without changing their definition, novel linguistic hedges are defined and used [Marín-Blázquez *et al.* 00, Liu *et al.* 01]. Part of the results of this thesis have been published [Marín-Blázquez & Shen 02a].

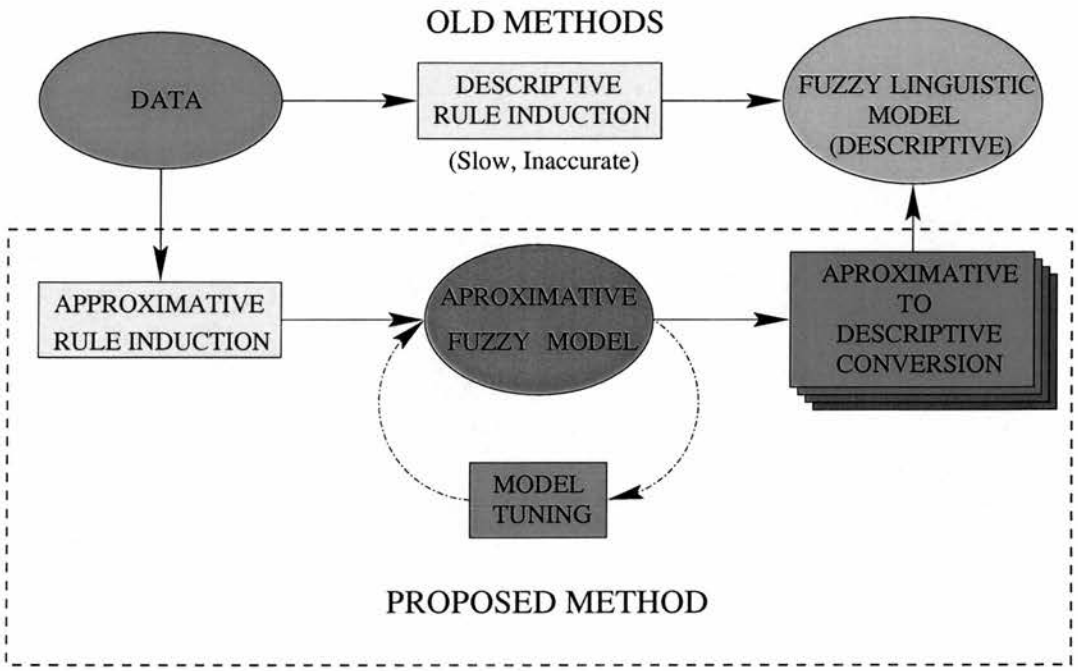


Figure 1.3: Descriptive fuzzy rule system generation proposed.

1.3 Structure of the Thesis

This section gives a brief account of the structural arrangement of the thesis.

1.3.1 Chapter 1: Introduction

This chapter has presented the reasons for performing this research. It has introduced the problems to be addressed and shown the main concepts such as fuzzy logic, approximative and descriptive fuzzy rules, and linguistic hedges. A proposal to solve the problem of generating descriptive fuzzy rules is outlined. In addition, this structure of the thesis is included.

1.3.2 Chapter 2: Background

This chapter covers two main sections, one about fuzzy theory and another about evolutionary computing. The fuzzy theory section includes basic concepts of fuzzy logic, fuzzy sets, and fuzzy inference. In addition, several types of fuzzy set, not only the ones to be used, are defined. The two fuzzy approaches, approximative and descriptive are also detailed. In the section on evolutionary computation, genetic algorithms, which are used to implement the present work are explained. Some details about other evolutionary techniques are also briefly reviewed.

1.3.3 Chapter 3: Automatic Generation of Fuzzy Rules

In this chapter most of the current techniques for automatic generation of fuzzy rules are reviewed. Distinct approaches to the generation problem are addressed, with a focus on the discussion of approximative and descriptive fuzzy modelling mechanisms. Finally, two specific algorithms, that will be used for experimental studies in the thesis, are introduced.

1.3.4 Chapter 4: Hedges

A description of the hedges used in this thesis is given in this chapter. The definitions of classical hedges are explained, and the newly proposed hedges explained. A novel method of carrying out the modifications that the hedges impose on the fuzzy sets is also developed.

1.3.5 Chapter 5: The Framework for Translation

This chapter deals with the main framework for the procedure to translate approximative models to descriptive ones. There, the methods for upgrading the training sets with new information needed for the translation are explained. The

objective functions used by the GA are shown. Finally, different strategies that can be followed to implement the translation procedure are given.

1.3.6 Chapter 6: Heuristic Methods

This chapter is devoted to the explanation of the working mechanisms of the heuristic methods used to obtain an initial first translation, for further search-based method to modify. A basic translation is illustrated with an example. The second part of the chapter explains several extensions of the basic heuristic method with suggestions about further improvements.

1.3.7 Chapter 7: The Genetic Algorithm

This chapter explains the specific genetic algorithm used in the present research. The chapter is divided with respect to the main components of a genetic algorithm. All the genetic operators are explained with graphic representations of the procedures. A section is devoted to the mechanism that is responsible for self-adapting assignment of the proportions in using different operators.

1.3.8 Chapter 8: Results on Benchmark Problems

Experimental proof of the effectiveness of the proposed translation mechanism is provided in this chapter. It starts with explaining the various problems to be used as the benchmark for comparative studies. In particular, it demonstrates that the new hedges proposed are better than the classical ones, and that the translated rule models are indeed of high transparency. Finally, modelling accuracy measures over different problems tested are shown and explained.

1.3.9 Chapter 9: Conclusions

This last chapter contains a summary of the achievements of this thesis. It also includes several suggestions about different lines of research that can be expanded in the future.

Chapter 2

Background

> "Everything is vague to a degree you do not realize till you have tried to make it precise."
>
> *Bertrand Russell*

2.1 Introduction

This chapter covers two main sections, one about fuzzy theory and another about evolutionary computing. The fuzzy theory section includes basic concepts of fuzzy logic, fuzzy sets, and fuzzy inference. In addition, several types of fuzzy set, not only the ones to be used, are defined. The two fuzzy approaches, approximative and descriptive are also detailed. In the section on evolutionary computation, genetic algorithms, which are used to implement the present work are explained. Some details about other evolutionary techniques are also briefly reviewed.

2.2 Sets and Logic

Sets are arbitrary groupings of elements. Such elements are said to be members of, or to belong to, a particular set. The elements can be anything of any imaginable kind,

from numbers to words, from people to atoms. They can be mixed in any way suitable for the purpose of the grouping, thus having sets that may include at the same time numbers, fruits, science fields, and anything. Nevertheless, sets are usually defined within a particular group of items, that is, the possible elements are restricted. The group of possible elements, not necessarily members, of a set is called its *domain*.

The number of elements that compose a set is called its *cardinality* and is represented mathematically by enclosing the name of the set in vertical bars. For instance, given a set named S with 10 elements, its *cardinality* $|S| = 10$. Sets can have infinitely many members. In some cases the elements of a set share a particular property, but that is not necessary. A set can be defined extensionally by enumerating all the members, for example, $A = \{1.3, Apple, Helium, Zoology, Edinburgh\}$, or can be defined intensionally by stating a property that all members share, like $A = \{x \mid x \in \mathbb{R}, x > 0\}$, the set of all positive real numbers, where $\mathbb{R}$ stands for the collection of all the real numbers. The only practical way of defining infinite sets is intensionally.

Traditional sets (also called "crisp" sets in the fuzzy literature) are sets where the members belong totally to the set. It is an all or none issue. This, of course, generates a crisp division line in the domain and divides it between members of that set and non-members of the set.

In particular, if the domain of a set is logic statements, and the set membership represent the things that are *true*, it is possible to divide all logical statements into true ones, that is, members of the set true, and not true ones (also called *false*), that is, non-members of the set true. Traditional logic stands over these two sets, true and false. Following this argument it is easy to see that the intersection (common members) of the sets true and false is empty (also called the law of contradiction, that is $A \cap \overline{A} = \emptyset$ with $\emptyset$ being the empty set) and that its union is the whole domain (the law of excluded middle or $A \cup \overline{A} = \mathcal{U}$ with $\mathcal{U}$ being the universe of discourse, the set of all possible elements).

2.3 Fuzzy Logic and Fuzzy Sets

Fuzzy Logic is an extension of Classical Logic introduced by Zadeh [Zadeh 65] in 1965. The two classic values of *true* and *false* are substituted by a real value between 0 and 1 representing the degree of certainty of a proposition, or the degree of membership to a fuzzy set. One element can, therefore, have partial membership to one or several sets.

So, a *Fuzzy Set* is a set in which its members can belong not in the classical way but in a certain degree ranging from total belonging (Membership value of 1) to not belonging at all (Membership degree of 0). Membership degree is usually denoted by the Greek letter μ.

To define a Fuzzy Set either the extensional method, or the intensional method can be used. The former enumerates its members and the membership degree of each one. The latter gives an expression called *Membership Functions* (denoted by $\mu_A(x)$) that, for each member x, return its membership degree to a given fuzzy set A.

A *Fuzzy Rule* is a special if-then rule that expresses imprecise dependencies between the variables of the antecedent and the variables of the consequent. The antecedent part is composed by a Fuzzy Logic expression. If such a Fuzzy Logic expression has a non-zero value then the rule is triggered. The consequent part can be a Fuzzy Set, a linear relation of the input variables, or a singleton value.

Such Fuzzy Rules are the base for *Fuzzy Modeling*. A Fuzzy Rule System consists of a set of fuzzy rules, the specifications (linguistic labelings and membership functions) of the fuzzy sets of the antecedents and consequents of the rules, and the evaluation method followed.

In the following sections all these concepts and items will be described in more detail.

2.3.1 Fuzzy Logic

Vagueness is older than fuzziness. First mentions to vagueness come from the philosopher/logician Charles Sanders Peirce in the late nineteenth century [Peirce 31]. Vagueness is associated to concepts with misty boundaries. Bertrand Russell introduced vagueness to logic [Russell 23]. Here, vague concepts are those that break Aristotle's law of non-contradiction, that is, in Aristotle's formulation: "One cannot say of something that it is and that it is not in the same respect and at the same time.", *A and not-A* do not hold simultaneously. Jan Lukasiewicz introduced logic operators for a multivalued logic, the min, max and negation operators [Lukasiewicz 70]. Max Black used the first fuzzy curves to describe fuzzy sets, although still used Russell wording and called them "vague sets". It was Zadeh [Zadeh 65] who first introduced the word fuzzy and developed a complete fuzzy set algebra. Zadeh did not refer to any prior work in the area like Lukasiewicz's, although Lukasiewicz's operators are the basic ones in fuzzy logic. This fact usually led to the confusion of assuming that multivalued logic started in the sixties with Zadeh's works although in truth it had started a century before.

Classical Logic fails to express some sentences about objects. The typical example is the sentence "John is tall". Tall is a concept that is not well defined. For example, if John is 1.80 meters he can be considered as truly tall among Pygmies but probably not among Masais. Avoiding such extremes, it can be hard to express the truthfulness of such a sentence among normal people. The most common answer to the question "Is John tall?" may well be "John is rather tall", an answer that does not fit in either yes or no.

At the same time classical logic also suffer from paradoxes such as the famous and ancient one of the liar from Crete. It goes as "A Cretan says that all Cretans lie". Is the Cretan telling the truth or not? If the Cretan says the truth then not all Cretans lie (as he, while being Cretan, is telling the truth) and therefore the sentence is not true

so he lied leading to a paradox. Likewise if the Cretan had lied, that is, the sentence is false, then he is telling the truth so the sentence is true! Bertrand Russell found that in set theory these paradoxes also appear (defining a set of all sets that do not include themselves, should such set be included in itself?). He tried to solve this problem with the theory of types but to no avail. As a result, in [Russell 23] he suggested to relax (basically to reject) Aristotle's law of excluded middle, therefore allowing membership of the same item to both a set and to its complementary. In the above case it allows the Cretan to tell the truth with 0.5 membership value and *not* telling the truth with 0.5 membership value as well.

Briefly, fuzzy set theory and fuzzy logics are extensions of classical logic and set theories. Fuzzy measure may be regarded as an extension of conventional probability but fuzziness does not mean probability. Fuzziness refers mainly to uncertainty and vagueness of human concepts. It should not be confused with probability or chances that a given event is true or false. A fuzzy description "$\mu_{\text{Intelligent}}(\text{Juanma}) = 0.9$" means that the concept of being intelligent is very appropriate to be applied to Juanma, not that he is very intelligent nine out of ten times and a fool one out of ten.

2.3.2 Fuzzy Sets

A fuzzy set is a set whose members can have a partial membership. This means that an atomic element (that is, an indivisible entity) is not necessarily completely inside a particular set. It can have a full membership, or a full non-membership, but this is not a must. They are called fuzzy sets because their boundaries are not clear but blurred while traditional sets have a sharp, crystalline, crisp boundary. If this fuzziness is applied to the true and false sets the above mentioned fuzzy logic emerges. The laws of excluded middle and non-contradiction are not true anymore as elements that are partially true are, at the same time, partially false.

Unlike in the classical set theory where an element is in a set or (but not and) in its complement, in Fuzzy Sets Theory, elements that may belong at the same time to a set

and to its complement. Therefore, Fuzzy Sets are the bricks upon which Fuzzy Logic is constructed. For example, John may belong to the tall set with a $\mu_{\text{Tall}}(\text{John}) = 0.7$ membership degree, but this implies that he also belongs (using classical Fuzzy Logic operators) to the set NOT tall (denoted by $\overline{\text{Tall}}$) with a $\mu_{\overline{\text{Tall}}}(\text{John}) = 0.3$ membership degree. So, in some way, he is and is not tall at the same time. Union, Intersection and Complement of fuzzy sets expressions are commonly defined as follows:

$$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\} \tag{2.1}$$

$$\mu_{A\cap B}(x) = \min\{\mu_B(x), \mu_B(x)\} \tag{2.2}$$

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{2.3}$$

It is easy to see from these that the union of a set and its complement is not the whole universe of discourse, and that the intersection of a set and its complementary is not the empty set $\emptyset$. So the laws of excluded middle and contradiction in classical set theory do not hold in fuzzy set theory. Figure 2.1 illustrates this.

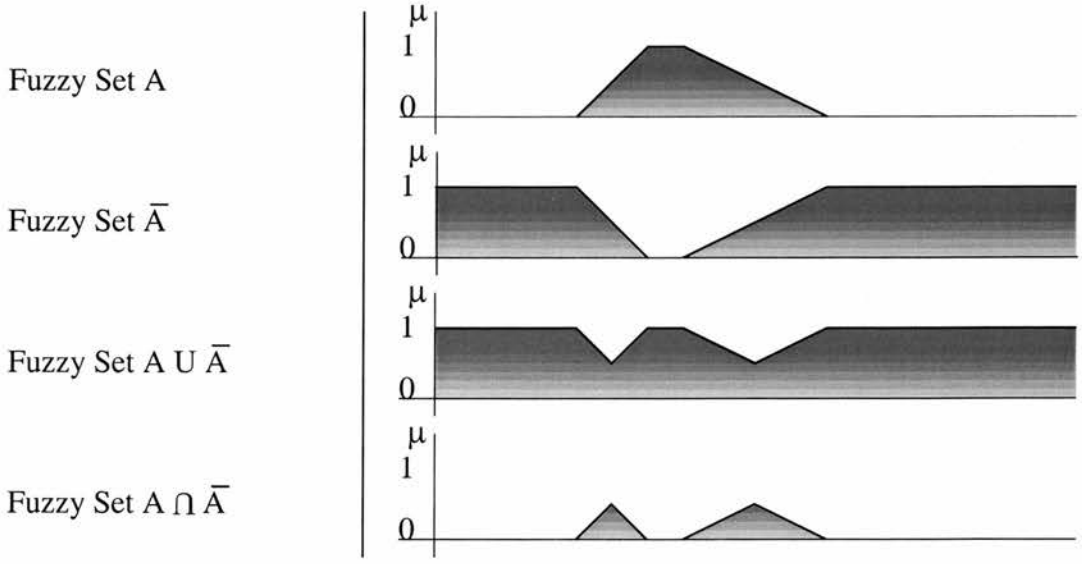


Figure 2.1: Union and Intersection of a fuzzy set and its complement

Again, there are two ways to express a fuzzy set. One is the extensional approach, where all members of a set are given. For Crisp Sets it is only needed to state which items are members. However, for Fuzzy Sets, for each member, its associated membership degree to the set also needs to be stated. In the literature such a set is given as pairs of membership degree/element $(A = \mu_A(x_1)/x_1, \mu_A(x_2)/x_2, \ldots, \mu_A(x_n)/x_n)$ [Tanaka 96]. The other way is intensional, where an expression is given about a property that all members of the set fulfill. For fuzzy sets such an expression gives the membership degree of each element and so it is a mapping from the domain X of the variable into the interval [0,1] ($\mu_A : X \rightarrow [0,1]$) forming the *membership function* of the fuzzy set under consideration.

2.3.2.1 One Dimensional Membership Functions

Thus a membership function is a function that associates each member to its membership degree for a given fuzzy set. There are infinite possibilities for such functions, but some of them are used more frequently in the literature (and in practice). They are usually named after the shape they have when a diagram is drawn. The most common are described below (with the domain considered restricted to $\mathbb{R}$):

2.3.2.1.1 Triangular (Figure 2.2)

They are characterized by three points. The point l where before which the membership value is zero and after which starts to increase its value linearly, the point c where it reaches its maximum value (usually 1) and starts decreasing linearly, and the point r where zero is reached again. Of course, $l \leq c \leq r$. The membership function expression (with centre membership value equal to 1) is:

$$\mu_{Triangular}(x) = \begin{cases} 0 & x \leq l \text{ or } x \geq r \\ \frac{x-l}{c-l} & l < x < c \\ 1 & x = c \\ \frac{r-x}{r-c} & c < x < r \end{cases} \tag{2.4}$$

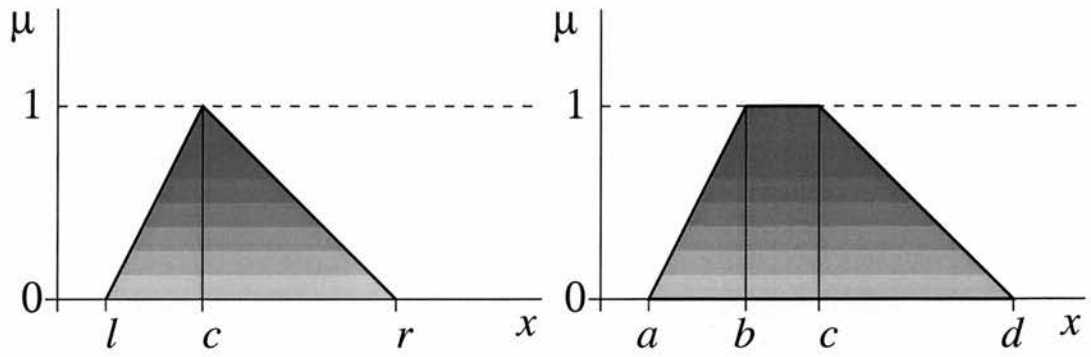


Figure 2.2: Triangular and Trapezoidal Fuzzy Set

2.3.2.1.2 Trapezoidal (Figure 2.2)

They are characterized by four points. The left down point a where it starts to rise, the left top point b where it reaches a membership value of one, the right top point c from where it starts to decrease, and the right down point d. Its expression is:

$$\mu_{Trapezoidal}(x) = \begin{cases} 0 & x \leq a \text{ or } x \geq d \\ \frac{x-a}{b-a} & a < x < b \\ 1 & b \leq x \leq c \\ \frac{d-x}{d-c} & c < x < d \end{cases} \tag{2.5}$$

2.3.2.1.3 Shouldered (Figure 2.3)

This type is characterized by two points and an indicator showing whether it is left or right shouldered. For a right shouldered fuzzy set the leftmost point a indicates where the increase from zero to full membership value starts, and the right point b shows where it reaches a membership value of one. The opposite applies for a left shouldered function. Their expressions are:

$$\mu_{LeftShouldered}(x) = \begin{cases} 1 & x < c \\ \frac{d-x}{d-c} & c \leq x \leq d \\ 0 & x > r \end{cases} \tag{2.6}$$

$$\mu_{RightShouldered}(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & x > b \end{cases} \tag{2.7}$$

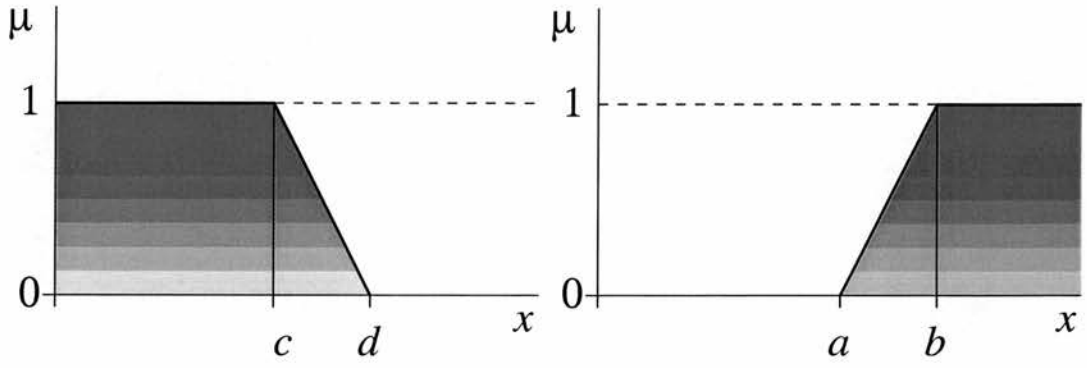


Figure 2.3: Left and Right Shouldered Fuzzy Set

2.3.2.1.4 Gaussian (Figure 2.4)

The following membership functions are tailed, that is, on the edges, the membership values never reach zero (but infinitely small). In particular the Gaussian membership function takes two parameters; c that represents the center of the fuzzy set (and the only point in which it reaches full membership) and σ that defines, the significant width of the fuzzy set. The expression that defines such a fuzzy set is:

$$\mu_{Gaussian}(x; c, \sigma) = e^{-\frac{1}{2}\cdot\left(\frac{x-c}{\sigma}\right)^2} \tag{2.8}$$

2.3.2.1.5 Bell (Figure 2.4)

The bell membership function is a more general version of bell-shaped functions. This tailored function is defined by three parameters. Again, c stands for the center of the fuzzy set (and again the only point of full membership), σ is also the width of

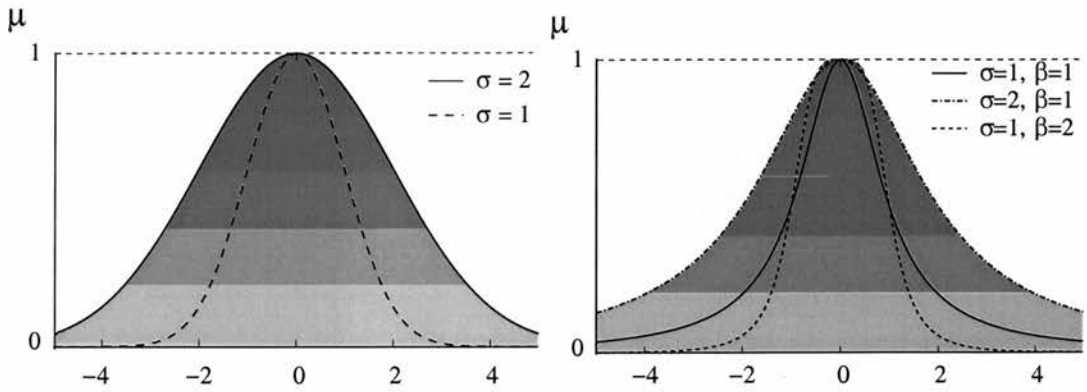


Figure 2.4: Gaussian (left) and Bell (right) fuzzy membership functions

the set, but a third parameter β is included. This latter parameter describes the slope that the curves which define the fuzzy set will have in the points of half membership. In particular, the slope in the rightmost of such two points is $-\frac{\beta}{2\cdot\sigma}$. If β is a negative number then the fuzzy set is an inverted bell (with zero membership in the center and approximately full membership in the infinites). In terms of description of the shapes, bell fuzzy membership function takes the same form as the Cauchy distribution that is used in the theory of probability and sometimes is referred as the Cauchy membership function. Note that this does not mean that a Cauchy fuzzy set represents probabilistic distributions, they just share the same expression:

$$\mu_{Bell}(x;c,\sigma,\beta) = \frac{1}{1+\left|\frac{x-c}{\sigma}\right|^{2\cdot\beta}} \tag{2.9}$$

2.3.2.1.6 Sigmoidal (Figure 2.5)

Sigmoidal membership functions, as shouldered ones, are mainly used for concepts that expand with full membership on one axis. A sigmoidal membership function is defined by two values: c represents the point where the fuzzy set attains *half* membership value. The second parameter β represents the slope at the point c. Positive values of β give "left shouldered" type of sigmoidal while negative values produce

"right shouldered" fuzzy sets. The expression that defines sigmoidal fuzzy sets is as follows:

$$\mu_{Sigmoidal}(x;c,\beta) = \frac{1}{1+e^{-\beta\cdot(x-c)}} \tag{2.10}$$

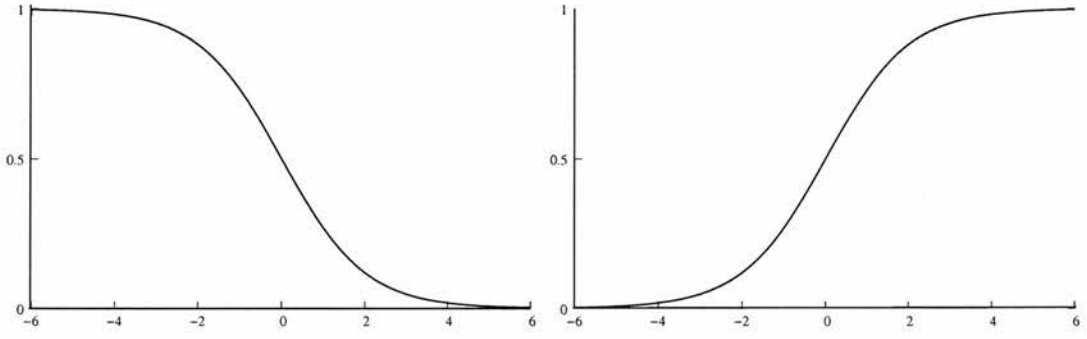


Figure 2.5: Sigmoidal with $c = 0, \beta = 1$ (left) and $c = 0, \beta = -1$ (right).

2.3.2.2 Multidimensional Membership Functions

There also exist fuzzy sets defined in multiple dimensions, maybe each defined in a different universe of discourse. In some cases the multidimensional fuzzy sets are created as extensions of lower dimension fuzzy sets or as projections of higher dimension fuzzy sets. In other cases they are obtained as combinations, via logical operators, of several lower dimension fuzzy sets. But of course, some of them can just be defined as an independent expression that involves all the inputs.

2.3.2.2.1 Projection of fuzzy sets

Projection allows multidimensional fuzzy sets to lose one (or more) dimensions. Projections are performed onto one (or more) axis. Being A a fuzzy set defined in $X_1, X_2, \ldots, X_n$ then its projection onto axis X_i with $1 \leq i \leq n$ is defined as:

$$p_i(A) = \int_{X_1,\ldots,X_{i-1},X_{i+1},\ldots,X_n} \sup_{x_i} \mu_A(x_1,\ldots,x_i,\ldots,x_n)/(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) \quad (2.11)$$

which again can be rewritten, by taking $B = p_i(A)$ as:

$$\mu_B(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) = \max_{x_i} \mu_A(x_1,\ldots,x_i,\ldots,x_n) \quad (2.12)$$

A graphical example of projection can be seen in figures 2.6 and 2.3.2.2.1. Figure 2.6 shows a bidimensional fuzzy membership functions and figure 2.3.2.2.1 shows projections in the x and y axis respectively.

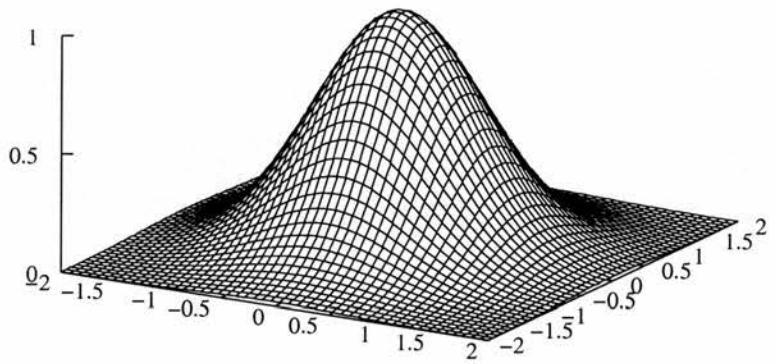


Figure 2.6: Bi-dimensional fuzzy membership function

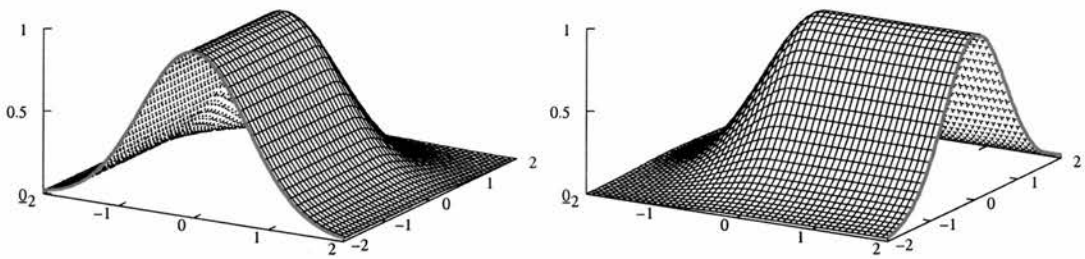


Figure 2.7: Projection (red line) on the x (left) and y (right) axis of the bi-dimensional fuzzy membership of figure 2.6

2.3.2.2.2 Composite fuzzy sets

Composite fuzzy sets are multidimensional sets that are created as the composition of several lower dimensional fuzzy sets combined through logical connectives. This means that the contribution of each dimension to the final fuzzy set can be isolated. This fact is very important because, as it will be seen in section 2.3.4.3, fuzzy rules are multidimensional fuzzy sets and, if they can be expressed as a composition of different one dimensional fuzzy sets, then interpretations of their meaning regarding each individual variable involved can be obtained. This fact is easy to see using the following equation. Assume that a given fuzzy set M is defined by a function of n different inputs, that is:

$$\mu_M(x_1,\ldots,x_n) = f(x_1,\ldots,x_n) \tag{2.13}$$

Suppose that the set M defines the antecedent part of a rule. Taking a look directly to f it may be very difficult to understand what particular values of the different variables affect the certainty of the rule. However, if the equation (2.13) can be rewritten as:

$$\mu_M(x_1,\ldots,x_n) = f(x_1,\ldots,x_n) = f_1(x_1)\ \mathrm{O}\ f_2(x_2)\ \mathrm{O}\ \ldots\ \mathrm{O}\ f_n(x_n) \tag{2.14}$$

with $O = \cup, \cap$ (or equivalently $O = OR, AND$) then the contributions of each variable to the final membership value can be easily spotted and interpreted.

2.3.2.2.3 Non-composite fuzzy sets (Figure 2.8)

Non-composite fuzzy sets are multidimensional sets whose membership functions can not be expressed as a composition of one dimensional fuzzy membership functions. If a fuzzy rule expression is non-composite then some of its dimensions are so intermingled that no individual information can be taken for them and they

effectively must be interpreted as a single entity. An example of non-composite bi-dimensional fuzzy set is shown in figure 2.8 where the following membership function is represented:

$$\mu_{NC}(x,y) = e^{-((x-y)^2+y^2)^2} \tag{2.15}$$

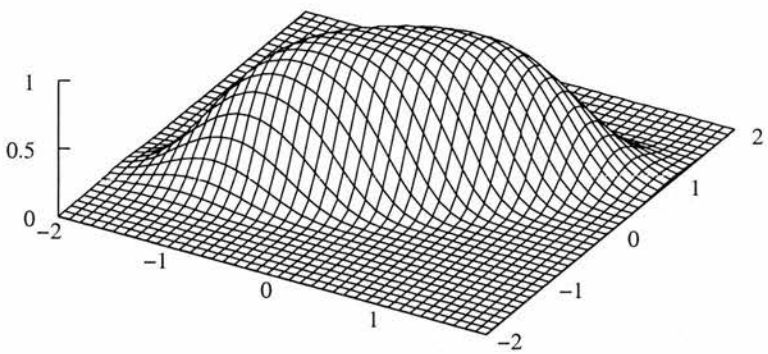


Figure 2.8: Non-Composite fuzzy membership function

Nevertheless, in many cases it is necessary to force mapping a multidimensional fuzzy set onto one dimensional ones. In particular, some of the often used automatic methods (see chapter 3) for fuzzy rule generation yield non-composite fuzzy membership functions as potential rules. In order to obtain decomposable rules (that is, rules that use one-dimensional fuzzy sets) a method to convert such non-composite into composite fuzzy sets must be performed. This is usually achieved by projecting the multi-dimensional fuzzy set into each of the components present in the expression. The rule is then built as the union of all the projections. Unfortunately, in the projection some information is usually lost and, therefore, when the relation is rebuilt some spurious solutions can be introduced and the original shape of the non-composite fuzzy relation can not be reobtained.

Figure 2.3.2.2.3 shows the fuzzy sets generated as the projection in the x and y axis of the fuzzy set shown in figure 2.8. The one-dimensional fuzzy sets obtained by such projection have the following membership function expressions:

$$\mu_{p_x(NC)}(x) = e^{-\left(\frac{x^4}{4}\right)} \tag{2.16}$$

$$\mu_{p_y(NC)}(y) = e^{-y^4} \tag{2.17}$$

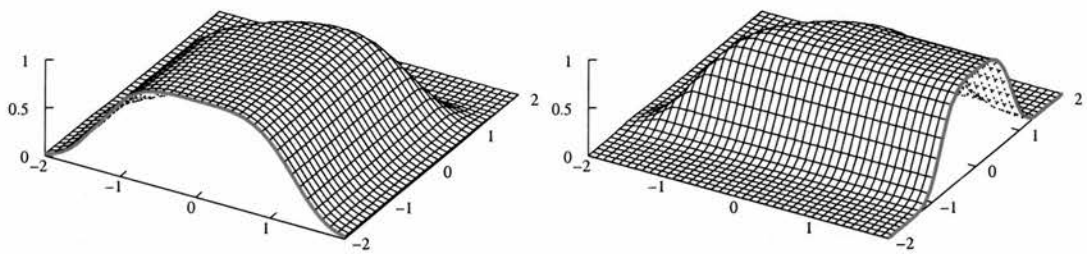


Figure 2.9: Projection (red line) on the x (left) and y (right) axis of the bi-dimensional fuzzy membership of figure 2.8

Figure 2.10 shows the shape of the rebuilt fuzzy set using two different operators for the union, the already mentioned maximum operator and the product (see section 2.3.3). The expressions that define both rebuilt fuzzy sets are:

$$\mu_{R_{\text{prod}}}(x,y) = e^{-\left(\frac{x^4}{4}\right)} \cdot e^{-y^4} = e^{-\left(\frac{x^4}{4}+y^4\right)} \tag{2.18}$$

$$\mu_{R_{\max}}(x,y) = \max\{e^{-\left(\frac{x^4}{4}\right)}, e^{-y^4}\} \tag{2.19}$$

Comparing the mathematical expressions (2.15) with (2.18) and (2.19), or figure 2.8 with figure 2.10, it is easy to see that they are not the same fuzzy sets. Another graphical example of this effect (from other perspective) can be also seen in figure 3.1.

2.3.3 Alternative Union, Intersection and Complement Operators

The Union, Intersection and Complement operators defined in equations 2.1, 2.2 and 2.3 are the classical operators given by Zadeh, but they are only one of many possible

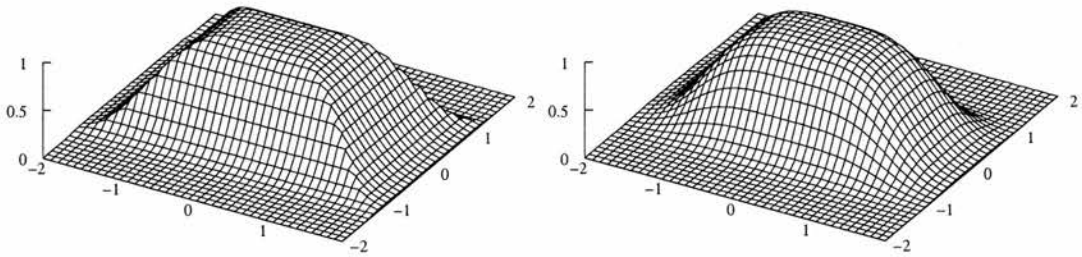


Figure 2.10: Rebuilt fuzzy membership function of figure 2.8 from the union of the projected fuzzy membership functions using min (left) and multiplication (right) as AND operator

consistent definitions. Although not used in this thesis, for the sake of completeness, the general definitions and some examples of alternative operators will be presented below:

2.3.3.1 Union, T-Conorm

Union of fuzzy sets is a function that aggregates membership values of two fuzzy sets. The general form of such a function is $S : [0,1] \times [0,1] \rightarrow [0,1]$:

$$\mu_{A \cup B}(x) = S(\mu_A(x), \mu_B(x)) = \mu_A(x) \breve{+} \mu_B(x) \tag{2.20}$$

Here, $\breve{+}$ is a binary operator, usually called T-Conorm or S-Norm, that must posses the following properties:

$$S(1,1) = 1, \qquad S(x,0) = S(0,x) = x \qquad \text{Boundary} \tag{2.21a}$$

$$S(x,y) \leq S(p,q) \quad \text{if } x \leq p \text{ and } y \leq q \qquad \text{Monotonicity} \tag{2.21b}$$

$$S(x,y) = S(y,x) \qquad \text{Commutativity} \tag{2.21c}$$

$$S(x,S(y,z)) = S(S(x,y),z) \qquad \text{Associativity} \tag{2.21d}$$

Examples of T-Conorms include:

$$\text{Maximum:} \qquad S(x,y) = \max\{x,y\} \tag{2.22}$$

$$\text{Algebraic Sum:} \qquad S(x,y) = x+y-x\cdot y \tag{2.23}$$

$$\text{Bounded Sum:} \qquad S(x,y) = \min\{1,(x+y)\} \tag{2.24}$$

$$\text{Drastic Sum:} \qquad S(x,y) = \begin{cases} x, & \text{if } y = 0 \\ y, & \text{if } x = 0 \\ 1, & \text{if } x,y > 0 \end{cases} \tag{2.25}$$

Figure 2.11 shows the effect of combining two trapezoidal fuzzy sets using different T-Conorms. In particular, from left to right and from top to bottom: the effects of Maximum, Algebraic Sum, Bounded Sum and Drastic Sum are illustrated.

Note that T-Conorms operators are sometimes interchangeably denoted by using $\cup$ or $\vee$ notation.

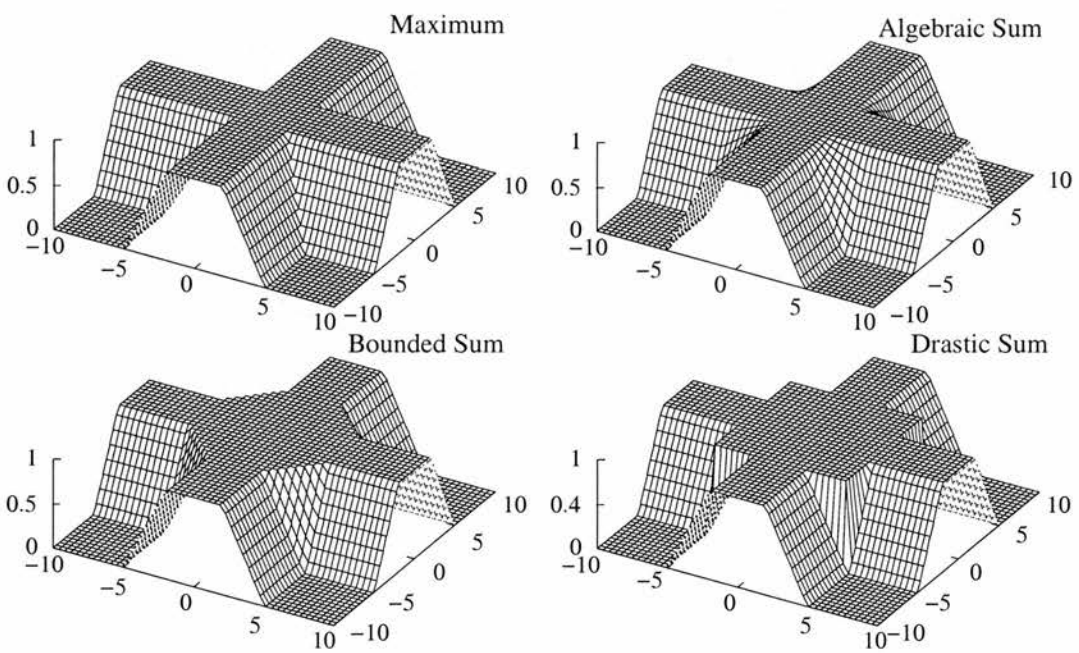


Figure 2.11: T-Conorms applied to combine two trapezoidal fuzzy sets

2.3.3.2 Intersection, T-Norm

The intersection of fuzzy sets is also a function that aggregates membership values of two fuzzy sets. As with union the general form of such a function is $T : [0,1] \times [0,1] \rightarrow [0,1]$:

$$\mu_{A \cap B}(x) = T(\mu_A(x), \mu_B(x)) = \mu_A(x) \ddot{*} \mu_B(x) \tag{2.26}$$

Likewise, $\ddot{*}$ is a binary operator called a T-Norm. T-Norm operators have to fulfill the following properties:

$$T(0,0) = 0, \quad T(x,1) = T(1,x) = x \qquad \text{Boundary} \tag{2.27a}$$

$$T(x,y) \leq T(p,q) \quad \text{if } x \leq p \text{ and } y \leq q \qquad \text{Monotonicity} \tag{2.27b}$$

$$T(x,y) = T(y,x) \qquad \text{Commutativity} \tag{2.27c}$$

$$T(x,T(y,z)) = T(T(x,y),z) \qquad \text{Associativity} \tag{2.27d}$$

Note that only the boundary property is different between T-Norms and S-Norms (or T-Conorms). Some examples of T-Norms can be:

$$\text{Minimum:} \qquad T(x,y) = \min\{x,y\} \tag{2.28}$$

$$\text{Algebraic Product:} \qquad T(x,y) = x \cdot y \tag{2.29}$$

$$\text{Bounded Product:} \qquad T(x,y) = \max\{0,(x+y-1)\} \tag{2.30}$$

$$\text{Drastic Product:} \qquad T(x,y) = \begin{cases} x, & \text{if } y = 1 \\ y, & \text{if } x = 1 \\ 0, & \text{if } x,y < 1 \end{cases} \tag{2.31}$$

Figure 2.12 shows the effect of combining two trapezoidal fuzzy sets using different T-Norms. In particular, from left to right and from top to bottom: the effects of Minimum, Algebraic Product, Bounded Product and Drastic Product are shown.

As with T-Conorms, note that T-Norms operators are sometimes denoted by using the $\cap$ or $\wedge$ notation.

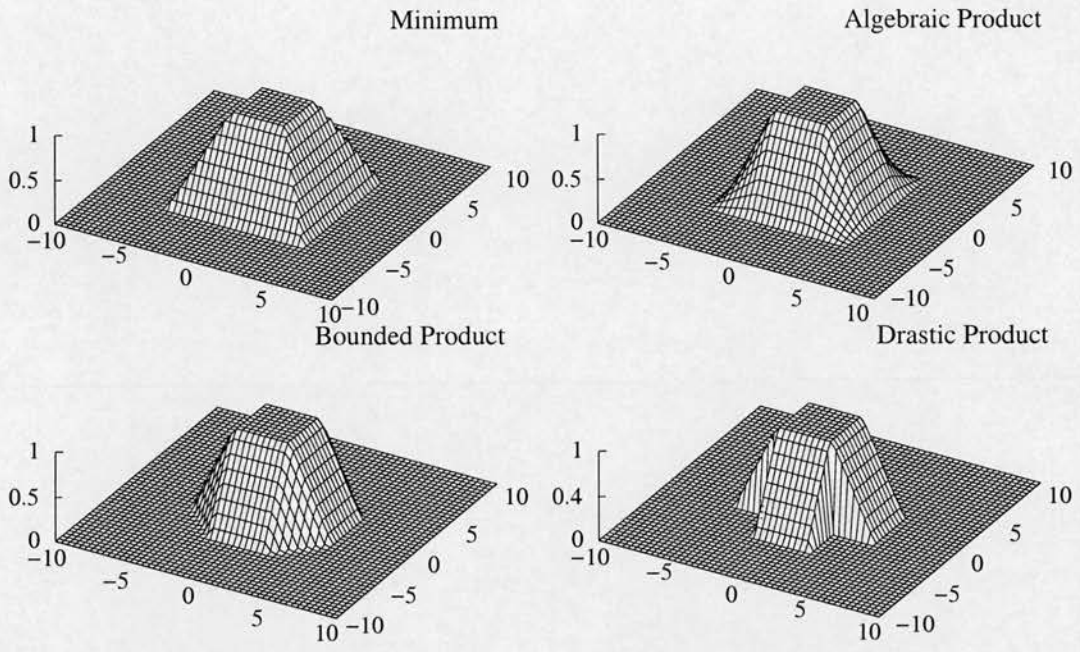


Figure 2.12: T-Norms applied to combine two trapezoidal fuzzy sets

2.3.3.3 Complement

Fuzzy Complement is a continuous function $N : [0,1] \rightarrow [0,1]$. Such a function must have the following properties:

$$N(0) = 1 \quad \text{and} \quad N(1) = 0 \qquad \text{Boundary} \qquad (2.32a)$$

$$N(x) \geq N(y) \quad \text{if } x \leq y \qquad \text{Monotonicity} \qquad (2.32b)$$

$$N(N(x)) = x \qquad \text{Involution} \qquad (2.32c)$$

In [Yager 79] and [Sugeno 77] families of fuzzy complements are defined. In particular, Yager's complement is defined as :

$$N_\delta(x) = \sqrt[\delta]{1-x^\delta} \tag{2.33}$$

with $\delta > 0$. Sugeno's complement is defined as:

$$N_\lambda(x) = \frac{1-x}{1+\lambda \cdot x} \tag{2.34}$$

with $\lambda > -1$. Clearly, the properties defined in equations 2.32a, 2.32b and 2.32c cover a whole set of complements. To force a complement to be the classical $N(x) = 1-x$ the following property is also needed:

$$\mu_A(x) + \mu_{\bar{A}}(x) = 1, \forall x \tag{2.35}$$

This implies that any change in the value of A is associated with a change on the value of $\bar{A}$.

2.3.3.4 Generalized DeMorgan's Law

A generalized version of DeMorgan's law is supported by a pair of T-Norm and T-Conorm together, with a complement operator. Using $T(\cdot,\cdot)$, $S(\cdot,\cdot)$ and $N(\cdot)$ notation the law can be expressed as follows:

$$S(x,y) = N(T(N(x),N(y))) \tag{2.36}$$

$$T(x,y) = N(S(N(x),N(y))) \tag{2.37}$$

or, if the symbols $\breve{+}$, $\breve{*}$ and $\bar{}$ are used, respectively, to denote a T-Conorm, T-Norm and fuzzy complement then it can be expressed as:

$$x \breve{+} y = \overline{\bar{x} \breve{*} \bar{y}} \tag{2.38}$$

$$x \breve{*} y = \overline{\bar{x} \breve{+} \bar{y}} \tag{2.39}$$

Therefore, by using DeMorgan's Law, for each T-Norm (or T-Conorm) the corresponding T-Conorm (or T-Norm) can be obtained through a given complement. In particular, using as complement $N(x) = 1 - x$ the T-Conorms and T-Norms given in equations 2.22 to 2.25 and 2.28 to 2.31 are, respectively, paired among themselves. These pairs are important to be able to resolve fuzzy inference (section 2.3.4).

2.3.3.5 Parameterized T-Norms and T-Conorms

Several authors have developed families of T-Norms and T-Conorms by parameterizing their definitions. The following are examples:

- [Schweizer & Sklar 63] with $-\infty \leq \alpha \leq \infty$:

$$\begin{aligned} T_{SS}(x,y;\alpha) &= \frac{1}{\sqrt[\alpha]{\max\{0, \sqrt[\alpha]{x} + \sqrt[\alpha]{y} - 1\}}} \\ S_{SS}(x,y;\alpha) &= \frac{1}{1 - \sqrt[\alpha]{\max\{0, \sqrt[\alpha]{1-x} + \sqrt[\alpha]{1-y} - 1\}}} \end{aligned} \tag{2.40}$$

- [Yager 80] with $\beta > 0$:

$$\begin{aligned} T_Y(x,y;\beta) &= 1 - \min\{1, \sqrt[\beta]{(1-x)^\beta + (1-y)^\beta}\} \\ S_Y(x,y;\beta) &= \min\{1, \sqrt[\beta]{x^\beta + y^\beta}\} \end{aligned} \tag{2.41}$$

- [Hamacher 75], with $\gamma > 0$:

$$\begin{aligned} T_H(x,y;\gamma) &= \frac{x \cdot y}{\gamma + (1-\gamma) \cdot (x + y - x \cdot y)} \\ S_H(x,y;\gamma) &= \frac{x + y + (\gamma - 2) \cdot x \cdot y}{1 + (\gamma - 1) \cdot x \cdot y} \end{aligned} \tag{2.42}$$

- [Sugeno 77] with $\delta \geq -1$:

$$\begin{aligned} T_S(x,y;\delta) &= \max\{0, (\delta + 1) \cdot (x + y - 1) - \delta \cdot x \cdot y\} \\ S_S(x,y;\delta) &= \min\{1, x + y - \delta \cdot x \cdot y\} \end{aligned} \tag{2.43}$$

- [Frank 79] with $\varepsilon > 0$:

$$\begin{aligned} T_F(x,y;\varepsilon) &= \log_\varepsilon\left[\frac{1 + (\varepsilon^x - 1) \cdot (\varepsilon^y - 1)}{s - 1}\right]] \\ S_F(x,y;\varepsilon) &= 1 - \log_\varepsilon\left[\frac{1 + (\varepsilon^{1-x} - 1) \cdot (\varepsilon^{1-y} - 1)}{s - 1}\right] \end{aligned} \tag{2.44}$$

- [Dombi 82] with $\lambda > 0$:

$$\begin{aligned} T_D(x,y;\lambda) &= \frac{1}{1+\sqrt[\lambda]{(\frac{1}{x}-1)^\lambda+(\frac{1}{y}-1)^\lambda}} \\ S_D(x,y;\lambda) &= \frac{1}{1+\frac{1}{\sqrt[\lambda]{\frac{1}{(\frac{1}{x}-1)^\lambda}+\frac{1}{(\frac{1}{y}-1)^\lambda}}}} \end{aligned} \tag{2.45}$$

- [Dubois & Prade 80] with $0 \leq \theta \leq 1$:

$$\begin{aligned} T_{DP}(x,y;\theta) &= \frac{x \cdot y}{\max\{x,y,\theta\}} \\ S_{DP}(x,y;\theta) &= \frac{x+y-x \cdot y-\min\{x,y,(1-\theta)\}}{\max\{1-x,1-y,\theta\}} \end{aligned} \tag{2.46}$$

In section 3 several automatic methods to obtain and tune fuzzy systems are explained. Some of these methods use parameterized T-Norms and T-Conorms for inference and tune the parameters to decrease error in the models.

2.3.4 Fuzzy Inference

In this section the main concepts concerning fuzzy inference are outlined, for a deeper explanation refer to the third chapter in [Jang *et al.* 97].

2.3.4.1 The extension principle

The extension principle [Zadeh 65, Zadeh 75] allows the transformation of any crisp function or mapping into a function that operates on fuzzy sets. This is of utmost importance as inference is a form of mapping. The extension principle works by assigning fuzzy memberships to the image values of the function based on the fuzzy membership of the original elements. Namely, it assigns to the image value the membership value of the original element. For example, in a function f defined from $f : X \rightarrow Y$ and A being a fuzzy set in X defined (extensionally) as:

$$A = \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \cdots + \frac{\mu_A(x_n)}{x_n} \tag{2.47}$$

then, the fuzzy set $B = f(A)$ defined in Y is:

$$B = \frac{\mu_A(x_1)}{f(x_1)} + \frac{\mu_A(x_2)}{f(x_2)} + \cdots + \frac{\mu_A(x_n)}{f(x_n)} \tag{2.48}$$

The above expression is correct for bijective functions (one-to-one correspondence mapping), that is, $f(x_i) = f(x_j) \rightarrow (i = j)$. In case of many to one mappings there would be several membership values for the same point in Y, that is, $\exists^* i, j$ with $i \neq j$ and $f(x_i) = f(x_j) = y$. In this case the membership value is:

$$\mu_B(y) = \bigvee_{x=f^{-1}(y)} \mu_A(x) = \underset{x=f^{-1}(y)}{T\text{-Conorm}}(\mu_A(x)) \tag{2.49}$$

Any T-Conorm will do here, the most common one being the maximum.

In case of n-dimensional functions $f : X_1 \times X_2 \times \cdots \times X_n \rightarrow Y$, the mapping from the n-dimensional universe given by the Cartesian product $X_1 \times X_2 \times \cdots \times X_n$ to the one-dimensional universe Y can be expressed as $f(x_1, x_2, \ldots, x_n) = y$, along with a series of n fuzzy sets $A_1, A_2, \ldots, A_n$ defined respectively in $X_1, X_2, \ldots, X_n$. Then, by the generalized extension principle, a fuzzy set B in Y is induced by the function f as:

$$\mu_B(y) = \begin{cases} \bigvee\limits_{(x_{1j},\ldots,x_{nj})=f^{-1}(y)} \left[\wedge_i \mu_A(x_{ij})\right] & \text{if } f^{-1}(y) \neq \emptyset \\ 0 & \text{Otherwise} \end{cases} \tag{2.50}$$

The above expression can be rewritten, using the T-Norm and T-Conorm notation, as:

$$\mu_B(y) = \begin{cases} \underset{(x_{1j},\ldots,x_{nj})=f^{-1}(y)}{T\text{-Conorm}} \left[T\text{-Norm}_i\, \mu_A(x_{ij})\right] & \text{if } f^{-1}(y) \neq \emptyset \\ 0 & \text{Otherwise} \end{cases} \tag{2.51}$$

In summary, the extension principle works by transferring the fuzzy membership of the original elements to the image values. For a n-dimensional vector of original

elements the membership value to be induced for the image is obtained as the intersection (intersection is done using a T-Norm) of the membership values of each component of the original element vector. In case of multiple memberships for an image then a union of the different values is done (union is performed using any T-Conorm paired with the above T-Norm).

2.3.4.2 Fuzzy Reasoning

Modus ponens is the traditional rule of inference in conventional logic. It states that a fact Y is true given a true fact X if there exists an implication $X \rightarrow Y$. That is:

$$\begin{array}{ll} \text{premise 1 : (Fact)} & X \text{ is } A \\ \text{premise 2 : (Rule)} & \text{if } X \text{ is } A \text{ then } Y \text{ is } B \\ \hline \text{consequence : (conclusion)} & Y \text{ is } B \end{array} \tag{2.52}$$

However, most of the real world facts involve a certain kind of fuzziness. To address this, the modus ponens is generalized in fuzzy logic such that:

$$\begin{array}{ll} \text{premise 1 : (Fact)} & X \text{ is } A' \\ \text{premise 2 : (Rule)} & \text{if } X \text{ is } A \text{ then } Y \text{ is } B \\ \hline \text{consequence : (conclusion)} & Y \text{ is } B' \end{array} \tag{2.53}$$

where A', B' are, respectively, fuzzy sets close (similar) to the fuzzy sets A, B. This is a specific version of the so-called compositional rule of inference [Zadeh 73], involving just two variables. The process of obtaining B' is referred to as fuzzy reasoning or approximate reasoning.

2.3.4.3 Fuzzy Rules

A fuzzy rule is an expression that represents imprecise dependencies between the variables that appear in its consequent part and the variables in its antecedent.

These parts are formed by the combination of membership degrees of the fuzzy sets defined in the domains of the relevant antecedent variables. Such a combination denotes a partition of the input spaces, associated with each partition is a consequent expression that could be a fuzzy set, a linear relation, or a singleton value. Therefore, the premise of a fuzzy rule indicates a fuzzy subspace of the input variables to which a relation with the output variables can be established [Sugeno & Kang 88].

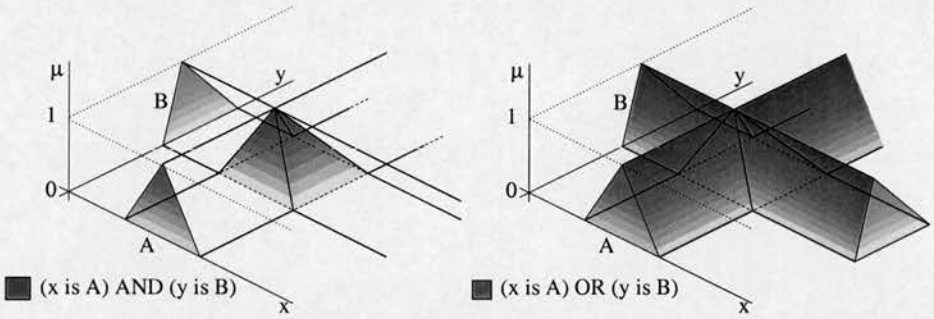


Figure 2.13: Fuzzy Partition induced by a Fuzzy Expression

Suppose that a k-dimensional input space $X_1 \times X_2 \times \ldots X_k$ and an m-dimensional output space $Y_1 \times Y_2 \times \ldots Y_m$ are given. In each input or output dimension a number, say, $l_{in\,i}$ or $l_{out\,j}$ with $i = 1,\ldots,k$ or $j = 1,\ldots,m$, of fuzzy sets are defined. For example, for the first input domain X_1 fuzzy sets $A_{11},...,A_{1l_{in1}}$ are defined. A typical Fuzzy Rule looks like:

$$\begin{aligned} R_h : \textbf{If } & x_1 \textbf{ is } A_1 \textbf{ and } x_2 \textbf{ is } A_2 \textbf{ and } \ldots \textbf{ and } x_k \textbf{ is } A_k \\ \textbf{then } & y_1 \textbf{ is } B_1 \textbf{ and } \ldots \textbf{ and } y_h \textbf{ is } B_m \end{aligned} \tag{2.54}$$

where x_i $(i = 1,\ldots,k)$ are the input variables and y_j $(j = 1,\ldots,h)$ are the output variables, and $A_h \in \{A_{h1}..A_{hl_{inh}}\}$, (i.e., a fuzzy set defined in the h-th domain of the input variables) and $B_j \in \{B_{j1}..B_{jl_{out\,j}}\}$.

As an example, figure 2.13 shows the partition induced in a two dimensional space by a fuzzy expression like x **is** A **and** y **is** B and one induced by the expression x **is** A **or** y **is** B.

In descriptive approaches [Delgado *et al.* 98b], each fuzzy set is labelled with a semantic meaning (linguistic interpretation) and the number of different fuzzy sets allowed in a model is restricted (by fixing the $l_{in}i$ and $l_{out}j$). Such sets and labels are normally given by an expert in the field. This way the rules extracted express a relation with semantic meaning and can be understood by humans. For example:

$$\begin{aligned} &\textbf{If } PH-Measure \textbf{ is } ACID \textbf{ and } Pressure-Measure \textbf{ is } HIGH \\ &\textbf{then } Situation \textbf{ is } DANGEROUS \end{aligned} \tag{2.55}$$

This expression is much more readable than equation 2.54 above. When such an expert is not present, the interpretation of the fuzzy sets involved may become a serious problem [Delgado *et al.* 98b, Sugeno & Yasukawa 93].

Nevertheless, what a rule represents is a fuzzy relation between the antecedent variables (inputs) and consequent variables (outputs). Assume A and A' are fuzzy sets defined in X, and B a fuzzy set of Y. Let the fuzzy implication $A \rightarrow B$ be defined as a fuzzy relation R defined in $X \times Y$. Using the compositional rule of inference an expression to calculate $\mu_{B'}(y)$ can be obtained as follows:

$$\begin{aligned} \mu_{B'}(y) &= \bigvee_x \left[\mu_{A'}(x) \wedge \mu_R(x,y)\right] \\ &= T\text{-}\underset{x}{\text{Norm}}\left[T\text{-Conorm}(\mu_{A'}(x), \mu_R(x,y))\right] \end{aligned} \tag{2.56}$$

In particular, using Mamdami's fuzzy implication [Mamdani & Assilian 75] and for just one antecedent condition and after a further simplification, equation (2.56) can be rewritten as (using only the AND, OR version):

$$\begin{aligned} \mu_{B'}(y) &= \left[\vee_x(\mu_{A'}(x) \wedge \mu_A(x))\right] \wedge \mu_B(y) \\ &= FS \wedge \mu_B(y) \end{aligned} \tag{2.57}$$

$\mu_{A'}(x) \wedge \mu_A(x)$ is the degree of compatibility between the fuzzy sets A' and A, which is also referred to as matching. FS denotes the degree of fulfillment of the antecedent

of the fuzzy rule, also known as the *firing strength* and, intuitively, it measures how compatible the fact is to the antecedent of the rule.

This can be extended to rules with several antecedents. For example, for two antecedents:"if x is A and y is B then z is C" (that can be seen as $A \times B \rightarrow C$) then the equation will appear as follows:

$$\begin{aligned}
\mu_{C'}(z) &= \vee_{x,y}\left[\mu_{A'}(x) \wedge \mu_{B'}(y)\right] \wedge \left[\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)\right] \\
&= \vee_{x,y}\left\{\left[\mu_{A'}(x) \wedge \mu_A(x) \wedge \mu_{B'}(y) \wedge \mu_B(y)\right]\right\} \wedge \mu_C(z) \\
&= \underbrace{\left\{\vee_x[\mu_{A'}(x) \wedge \mu_A(x)]\right\}}_{FS_1} \wedge \underbrace{\left\{\vee_y[\mu_{B'}(y) \wedge \mu_B(y)]\right\}}_{FS_2} \wedge \mu_C(z) \\
&= \underbrace{(FS_1 \wedge FS_2)}_{\text{Firing Strength}} \wedge \mu_C(z)
\end{aligned} \tag{2.58}$$

A graphical example of the above equation (2.58) can be seen in figure 2.14. The green small pyramid represents the evidence obtained applying the Generalized Modus Ponens with the rule "if x is A and y is B then z is C" and the facts represented by A' and B'.

Likewise, in case of having several rules, for example "if x is A_1 and y is B_1 then z is C_1" and "if x is A_2 and y is B_2 then z is C_2" the final $\mu_{C'}(z)$ would be:

$$\begin{aligned}
\mu_{C'}(z) &= \left(\underbrace{\left\{\vee_x[\mu_{A'_1}(x) \wedge \mu_{A_1}(x)]\right\}}_{FS_{11}} \wedge \underbrace{\left\{\vee_y[\mu_{B'_1}(y) \wedge \mu_{B_1}(y)]\right\}}_{FS_{12}} \wedge \mu_{C_1}(z)\right) \vee \\
&\vee \left(\underbrace{\left\{\vee_x[\mu_{A'_2}(x) \wedge \mu_{A_2}(x)]\right\}}_{FS_{21}} \wedge \underbrace{\left\{\vee_y[\mu_{B'_2}(y) \wedge \mu_{B_2}(y)]\right\}}_{FS_{22}} \wedge \mu_{C_2}(z)\right) \\
&= \left\{ \underbrace{(FS_{11} \wedge FS_{21})}_{\text{Firing Strength Rule 1}} \wedge \mu_{C_1}(z)\right\} \vee \left\{ \underbrace{(FS_{21} \wedge FS_{21})}_{\text{Firing Strength Rule 2}} \wedge \mu_{C_2}(z)\right\} \\
&= \mu_{C'_1}(z) \vee \mu_{C'_2}(z)
\end{aligned} \tag{2.59}$$

Equation (2.59) says that when multiple rules are involved, the resultant fuzzy set is the union (aggregation) of the fuzzy sets obtained from applying each rule individually. That is the reason that fuzzy rule systems are also treated as additive systems [Dickerson & Kosko 96].

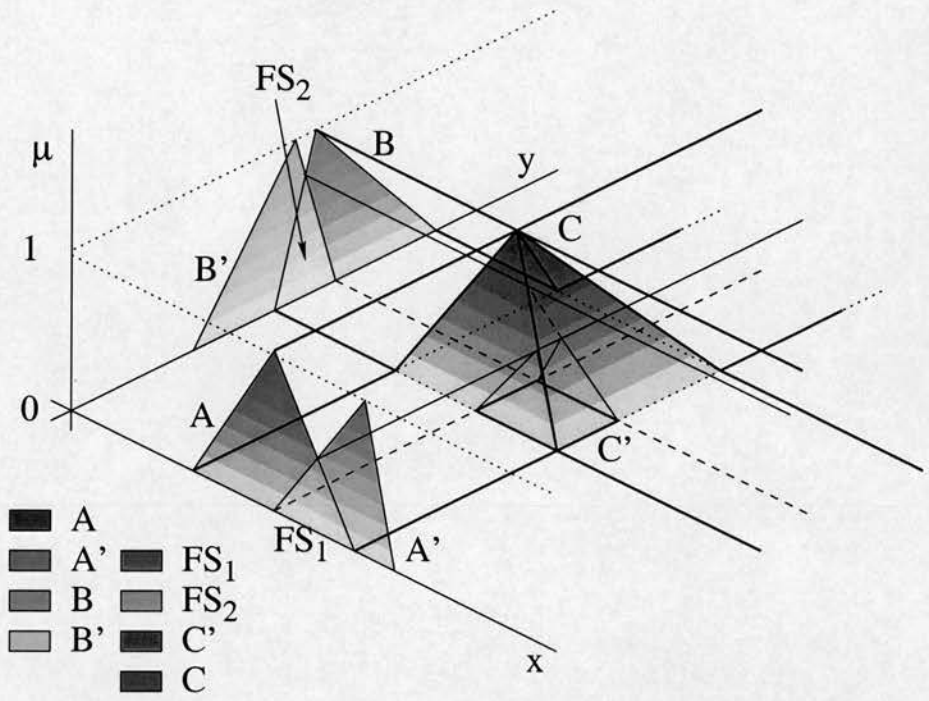


Figure 2.14: Fuzzy Reasoning for a rule with two antecedents

2.3.5 Fuzzy Modeling

Fuzzy modeling is an approach used to form a fuzzy system model. It is based on the idea of finding a set of local input-output relations describing the system to be modelled. A fuzzy model consists of a number of fuzzy if-then rules and the definitions of the fuzzy sets that appear within such rules.

In fuzzy modeling, the most important problem is the identification process which consists of two aspects: structure identification, and parameter identification [Sugeno & Yasukawa 93]. The structure identification consists of finding the input attributes which affect the output from a collection of possible attributes, and the identification of an optimal number of fuzzy partitions of the input space. Usually the number of rules depends on it. The parameter identification is concerned with finding the optimal parameters of the fuzzy sets membership functions that are used in the fuzzy rules.

Therefore, the task of fuzzy modeling is to find a finite set of fuzzy rules capable of reproducing the input-output behavior of the system being considered. For classification problems, without losing generality, the system to be modeled is assumed to be a MISO (Multi-Input Single-Output) one. That is, a system of M inputs and one output that can be described by a set of K rules such as:

$$R_i : \text{If } x_1 \text{ is } S_i^1 \text{ and } ... \text{ and } x_M \text{ is } S_i^M \text{ then } y \text{ is } Set\ \mathrm{O}/Class_h/Value/Expression$$

where R_i is the ith rule ($1 \leq i \leq K$), x_j is the jth input variable ($1 \leq j \leq M$), y is the output variable to be assigned to one of the possible output values or classes, and S_i^j are fuzzy sets for these variables. S_i^j can be either a labelled and previously defined (thus descriptive) fuzzy set or an unlabelled (and sometimes unlabellable) membership function that defines itself a fuzzy set (hence approximative). Descriptive fuzzy sets are human defined and fixed throughout both the modeling and the inference processes (though minor micro-tuning [Marín-Blázquez & Shen 02b] of the definition may be permissible).

For classification applications the inference is done by selecting the rule that fires with highest strength. This firing strength is calculated as a T-Norm of the result of the individual matches of each variable in the antecedent (degrees of compatibility). For function modeling an aggregation of all fired rules is done by an S-Norm of the output values.

The majority of automatic rule generation methods follow the general principle of supervised learning [Mitchell 97]. The only information about the behaviour of the system being modeled is assumed to be a (usually large) set of input-output example pairs, where for each instantiation of the input variables an associated class or value is indicated:

$$\Omega = \{(x_{t1}, x_{t2}, \ldots, x_{tM}, y_t)\} = \{(x_t^M, y_t), t = 1, \ldots, N\}$$

The model to be generated is required to be able to approximate the function $\varphi : X^M \rightarrow Y$ (that theoretically underlies the system behavior) in the way most consistent

with the given examples of inputs-output pairs. Clearly, the collection of the data examples is presumed to represent the system behavior in the product space $\left(X^M \times Y\right)$, where $X^P = (X_1 \times X_2 \times \ldots \times X_M)$, and $X_1, X_2, \ldots, X_P$ are the domains of discourse of the inputs while Y is the domain of the output.

In the descriptive fuzzy approach (see section 2.3.7 for details) the number of fuzzy sets and a preliminary definition of them is already given. Furthermore, such fuzzy set definition is fixed and can not be modified to optimize it. Usually the structure identification is done by using an algorithm that tries to extract the smallest subset possible of the Cartesian product of all sets defined in each dimension of the joint input and output space. The set of all possible rules will, in general, have

$$R : l_{in}1 \times l_{in}2 \times \cdots \times l_{ink} \times l_{out}1 \times \cdots \times l_{out}m \tag{2.60}$$

possible different rules. For example, the rule induction algorithm [Lozowski *et al.* 96, Shen & Chouchoulas 00] extracts from such Cartesian product a set of rules that comply with a tolerance error ε. An extensive explanation of this method will be provided in Section 3.5.

The main problem of this approach is the combinatorial explosion if there are a large number of input or output variables to consider. In order to reduce the number of such variables, a Principal Component Analysis [Haykin 94, Ross & Hallam 96] or Rough Sets Attribute Reduction [Pawlak 91, Jelonek *et al.* 95, Shen & Chouchoulas 99] or any other method of extraction of the most important input variables can be applied to the data. Nevertheless, this approach is, for problems with more than fifteen input variables and each having a reasonable number of fuzzy labels, too computationally expensive to be useful.

Another very important point to be taken into account in fuzzy rule systems is the way in which the input space is divided. In particular three different families can be identified, namely:

- Grid Partition:

  A grid partition is the result of the Cartesian product of a strict fuzzy partition in each of the variables of the system to be modeled. Here, a strict fuzzy partition P of a universe of discourse X is a group of fuzzy sets S_i defined on X that possess the following properties:

$$\forall x \in X, \exists S_i \in P \mid \mu_{S_i}(x) > 0 \tag{2.61}$$

$$\forall S_i \in P, \exists x \in X \mid \mu_{S_i} = 1 \tag{2.62}$$

$$\forall x \in X,\ \exists S_i, S_j \in P \mid \mu_{S_i}(x) + \mu_{S_j}(x) = 1,\ \forall S_k \in P, k \neq i, k \neq j, \mu_{S_k}(x) = 0 \tag{2.63}$$

  The first property ensures coverage. The second one provides normality (all sets have a point with a full membership). The last property effectively forces a point to belong, at most, to only two different sets.

  The fuzzy rule system is composed by all the rules of the grid. The coverage of the input space is therefore total. This type of fuzzy system has been extensively used in control as most of them use only a few state variables to perform the control task. The advantages of these systems come from the complete coverage and the easiness of its ruleset generation (just using the Cartesian product of the different sets). On the other hand, as the number of variables and fuzzy sets in the partitions increases the number of rules explodes exponentially, resulting in the so-called "curse of dimensionality" [Ishibuchi *et al.* 97, Dick *et al.* 99]. This inconvenience is partly solved in other types of partition. An example of how grid partition covers a bi-dimensional input space can be seen in figure 2.15.

- Tree Partition:

  In this partition type the coverage of the input space is again ensured. In this case each zone of the space can be reached after traversing a decision tree. The

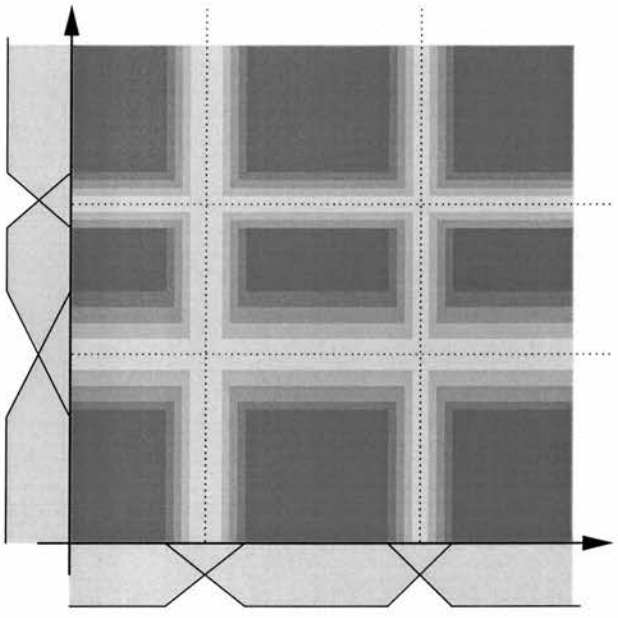


Figure 2.15: Fuzzy Grid Partition

usual way of obtaining these partitions is by iteratively dividing by two regions of the input space. This assumes, without losing generality, binary trees. The definition can be trivially extended to n-ary trees Each division is performed in, and generates, a node of the tree. Figure 2.16 shows a sequence in the generation of such a partition on a bi-dimensional input space, with colored codes for the depth in the tree. Blue represents the first level, red the second, green the third and brown the fourth. Each final cell represents an independent rule.

- Scattered Partition:

  In this last partition type each rule is independent of the others. Coverage is not ensured and usually the rules cover only the interesting parts of the input space. In most of its applications there are vast extensions of the input space that are empty. Grid partitions and to some extent tree partitions may include rules attempting to cover these zones and hence may be wasteful for certain tasks. Scattered partitions usually make better use of the rules available. However,

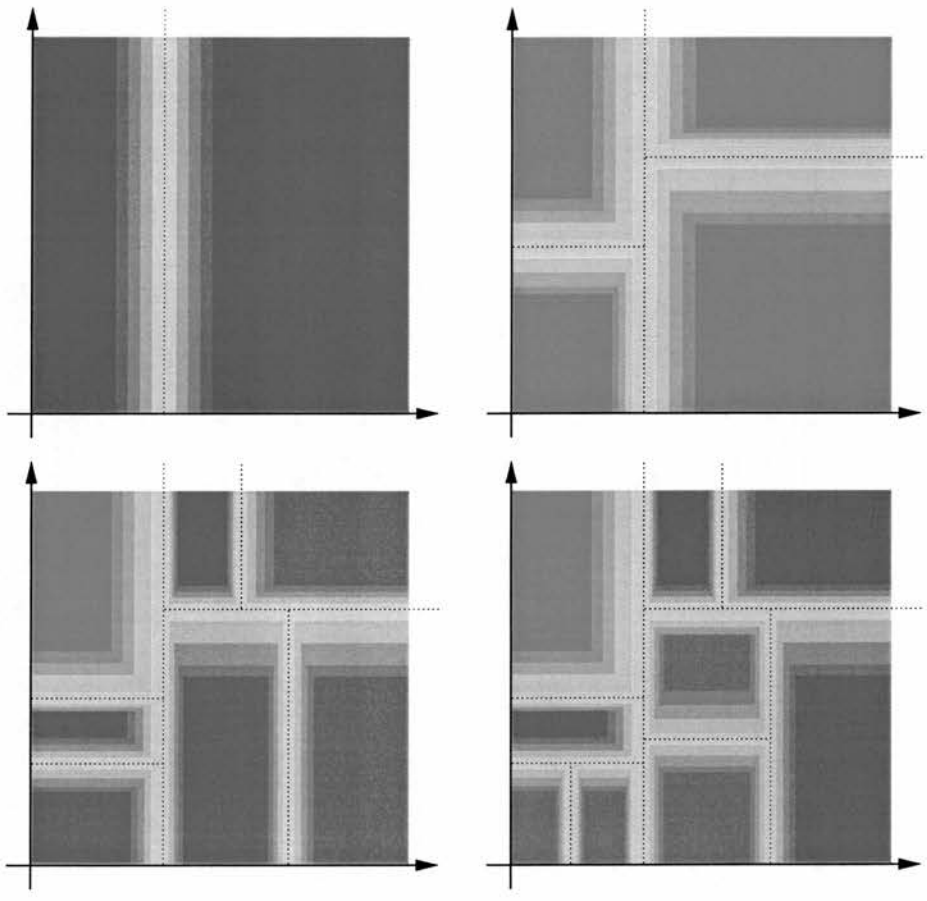


Figure 2.16: Fuzzy Tree Partition

this type of partition may suffer from the fact that the overlapping of rules is encountered more often, with the risk of inconsistencies in the rule base. It is also potentially dangerous to keep zones of the input space without coverage as some unforeseen or noisy data can fall in such zones when the generated ruleset is in action. To overcome this weakness default outputs or some more complex systems (such as GEM rules in [Marín-Blázquez *et al.* 99]) have been proposed.

In figure 2.17 an example of scattered partition on a bi-dimensional input space is represented. It is easy to recognize the large uncovered zones and the more effective use of fewer rules.

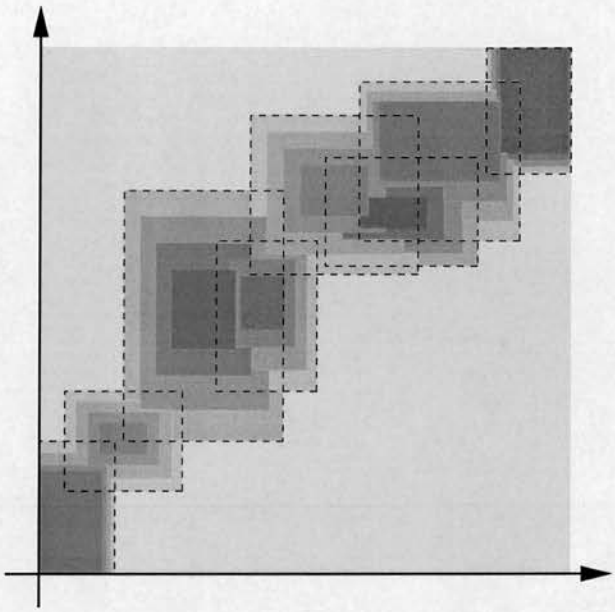


Figure 2.17: Fuzzy Scattered Partition

2.3.6 Approximative Approach

As seen, fuzzy theory has a sound mathematical background. Fuzzy inference can be performed using the mathematical expressions shown in the previous sections. Rules can be constructed and used to model systems where no mathematical definitions are available up to any degree of precision needed. It has been proven that fuzzy rule systems are universal approximators [Wang 92, Kosko 94]. However, to achieve this the fuzzy sets are usually chosen and defined to best fit the data to be modeled. This almost certainly deprives the fuzzy sets of any chance of representing human concepts about the universe of discourse on which they are defined. The tuning mechanisms that are usually applied tend to distort the original definition of fuzzy sets into very obscure shapes. Even the operators used are sometimes modified (always inside the constraints given by the properties that T-Norms [equations (2.27)] and T-Conorms [equations (2.21)] must possess), making the task of interpreting the inference process itself even more difficult.

In short, approximative fuzzy systems are much more concerned with precision than with interpretability.

Sigmoid and radial basis functions have been rather popular in approximative approaches. These functions have certain properties that make them ideal for certain optimization tasks. The most significant properties are:

- Infinite tails: This means that the value of the membership function will never be zero except at $\pm \infty$. This has a positive side effect of generating a complete coverage of the domain space. Any partition of the domain space using sigmoid fuzzy sets will cover all possible values regardless the particular sets (although for some values the membership can be negligible). An example can be seen in figure 2.18.

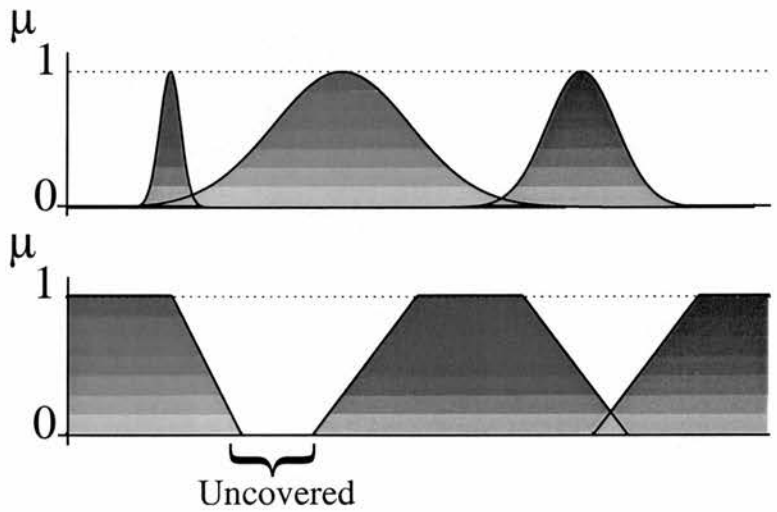


Figure 2.18: How sigmoids ensure complete coverage of the variable domain no matter how the sets are defined while piecewise fuzzy sets can not guarantee this.

The fact that, in some cases, the coverage can be almost negligible in terms of the value of membership function does not prevent rules from being fired (even slightly) and, therefore, extracting conclusions. Nevertheless, in these cases the evidence to support such conclusions may be minimal.

Infinite tails also means that *all* rules of a fuzzy system will fire (as all points belong to all fuzzy sets defined in their domain, however minimal their membership value).

- Derivable: Differentiable functions are needed for gradient descent optimization algorithms. In particular, the most popular is the back-propagation [Rumelhart *et al.* 86a, Rumelhart *et al.* 86b] algorithm applied in most neurofuzzy systems. The expressions for the derivatives of the fuzzy sets explained in section 2.3.2 can be found in chapter 2, section 4.2 of [Jang *et al.* 97].

The first of these properties may be a source of criticism for approximative systems. Most of the fuzzy systems are additive, that is, the output comes from the aggregation of the evidence collected among all fired rules. This is not in itself something undesirable, on the contrary, the aggregation of different pieces of evidence can be a great advantage. The smooth interpolations obtained using fuzzy rules, very appreciated in control, are precisely a result of additive systems. The problem arises when too much different and potentially contradictory evidence is collected. Too many fired rules mean that it may be difficult to assign credit to a small enough group of rules so that the justification of taking a decision is clear. In other words, it is difficult to understand why a result has been obtained if a large number of rules are involved in the process of applying a single inference step. Tailed functions, as mentioned previously, cause all the rules to fire, leading to the extreme of this problem. The most common way to tackle this is by setting a minimum firing strength threshold to be considered, but then the full coverage is lost. Nevertheless, not many authors of approximative systems claim transparency of their systems, in which case the number of fired rules is no longer an important issue.

Approximative approaches are the most commonly adopted and many such techniques for ruleset generation and tuning have been developed. In fact, each

technique turns a bit more the knot in a frantic race to obtain the highest precision at the least computational cost. For a summary of the main generation techniques see chapter 3.

2.3.7 Descriptive Approach

Descriptive fuzzy sets are linguistically labeled fuzzy sets. Such sets are human defined and, in principle, are not allowed to be changed either in the rule generation process or in the tuning process of the resulting ruleset. Each domain of each variable involved is partitioned by humans into several fuzzy sets. This partition should maintain several desirable properties [Valente deOliveira 99], such as coverage (that all points of the domain are covered at least by a set), distinguishability (that the sets are to be quite different among themselves), and normality (there should be, for each set, some points with a full membership, a genuine example of a particular concept). As humans can not process efficiently more than 7 ± 2 different entities in the short term memory [Miller *et al.* 60], a linguistic variable of practical use should not have more than such a number of possible different labels.

For classification, a MISO system can be described by a set of K rules such as:

$$R_i : \text{If } x_1 \text{ is } D_i^1 \text{ and ... and } x_M \text{ is } D_i^M \text{ then } y \text{ is } Class_h \tag{2.64}$$

where R_i is the ith rule ($1 \leq i \leq K$), x_j is the jth input variable ($1 \leq j \leq M$), y is the output variable to be assigned to one of the possible output classes, and D_i^j are descriptive fuzzy sets for these variables. D_i^j can be either a single descriptive fuzzy set or a combination of one or two hedges and a descriptive fuzzy set. Note that more than two hedges per variable are allowed in theory. However, a joint use of more than two hedges often destroys the readability of the resulting descriptive rules and hence is not desirable to be employed in practice.

It is, in general, computationally prohibitive to perform exhaustive search in the space of all possible combinations of descriptive sets (with or without the use of

linguistic hedges), in order to obtain descriptive rules. This is due to the "curse of dimensionality" [Ishibuchi *et al.* 97, Dick *et al.* 99]. The higher the number of variables available, and/or the higher the cardinality of the fuzzy partition for each variable, the higher (exponentially) the number of possible rules. This is generally true for approximative models as well. In fact, the size of all possible rules associated to the general product ruleset R_{Set} of a given system, allowing up to two hedges per fuzzy set for example, may be represented by:

$$|R_{Set}| = H_{L_1 1} \times H_{L_1 2} \times L_1 \times \ldots \times H_{L_M 1} \times H_{L_M 2} \times L_M \times S = h^{2\times M} \times s \times \prod_{i=1}^{p} k_i \quad (2.65)$$

where L_i is the fixed set of labels for the ith variable in the input space, of a cardinality $|L_i| = k_i$; $H_{L_i j} \in H$, $j = 1,2$, with H being the set of applicable hedges of a cardinality $|H| = h$; and S the fixed set of class labels of the output space, of a cardinality $|S| = s$.

Of course the modeling task is to select the smallest possible subset of R_{Set} that characterizes the system under consideration to as high a degree as possible. This value increases dramatically as input dimensionality increases. This makes it impossible to perform exhaustive search for any moderately sized problem. Even robust but non-exhaustive search techniques such as genetic algorithms [Holland 75, Goldberg 89], may not perform well when $|R|$ is large [Ishibuchi *et al.* 99]. This is mainly because many of the rules generated may not cover any data. The subset of interesting rules that at least cover some data may indeed be very small compared with the total; much effort of the search is often wasted in order to find that small subset. However, the search of interesting rules can be considerably reduced if it starts with a rough solution and then, say, a GA is used to optimize this solution.

The question becomes how to find a rough solution efficiently. Fortunately, there already exist approximative rule generation techniques that are, focusing on given data, able to find very accurate rules, without using pure and brute force search. Being data-driven [Mitchell 97], these techniques avoid the empty parts of the input space. There are no restrictions over the fuzzy sets they use, that is, these sets do not have

to satisfy any prescribed linguistic interpretation. Furthermore, the fuzzy sets used in the antecedent do not have to be chosen among a predefined number of sets but they can be different among each premise of each rule. The resultant approximative rules "point" to places in the search space where desirable descriptive rules potentially exist. These approximative rules can then be transformed into descriptive ones with a heuristic method, which is used as the generator of the first population for an optimizer like a GA that will then fine-tune the translation.

Figure 1.3 has shown the basic ideas of the present work. Instead of using a direct descriptive rule induction technique, which is generally rather slow and inaccurate, it is proposed to use a fast and accurate approximative rule induction algorithm first. (The approximative fuzzy model induced can be tuned to improve its modeling accuracy.) The approximative model is then converted into a fuzzy linguistic model that utilizes predefined descriptive sets. It is this conversion process that forms the major work described herein.

Through the use of approximative rules, a vast volume of the search is already done. The effort of the optimizer (GA) is directed to perform fine adjustments. The emerging solution, i.e. a descriptive ruleset, can be improved due to the neighborhood search operators included in the GA or inserted between GA generations. In general, the better the initial translation the less effort the GA will have to apply, as the initial solution is already close to the final ruleset, usually far closer than starting with a random one.

2.4 Evolutionary Algorithms

As pointed out in [Luger & Stubblefield 97], one of the aspects of AI's famous Physical Symbol System Hypothesis is that problems are solved by searching among alternative choices. Optimization can be viewed as a search process. Any general textbook of AI has a part devoted to search, in fact it is an important issue in AI.

Amongst different search strategies, heuristic search is a type of search guided towards (somehow measured) promising lines of search.

A particular approach for heuristic search is known as "Neighborhood Search". This type of search is based on a basic schema of starting with a solution and examining the vicinity of it in the hope of finding a better solution. Within this type of search, Evolutionary Computing (Evolutionary Computing with Fuzzy Systems, and Artificial Neural Networks, is sometimes referred to as Computational Intelligence or Soft Computing) is one that is commonly applied. Another is the so-called Hill Climbing. For an introductory overview of these consult [Tuson 98].

Hill Climbing is based upon the idea of moving always in a direction that improves the actual solution, or allowing the search to "climb the hill" to find a better solution that lies in the top of the possible solution landscape. Evolutionary Computing is based upon the works of Charles Darwin [Darwin 59], applying his idea of evolution by natural selection to the world of search. A "population" of solutions compete, mate and mutate, with only the "best" solutions surviving to the next generation (the most promising population members). That is, Evolutionary Techniques cover a group of problem solving strategies characterized by the use of a population of solutions that evolve in a analogous way to biological organisms. The way they evolve is by interaction of the solutions or by individually manipulating members. The new members so created will replace older or worse members of the population, moving (hopefully) to a better solution.

These techniques have the advantage of considering many different solutions at the same time, as each member can be considered as a position from where the system is searching. As the best (or more promising) solutions often remain in the population, the system will continue searching only in those promising directions. Poor members represent bad, or unpromising solutions, and they are usually discarded and replaced by better offspring. In addition, these techniques rely on the idea that mixing chunks of good solutions together may result in better solutions. This is the idea behind crossover.

There are several research pathways in Evolutionary Computation, and this thesis will use one approach from this field: Genetic Algorithms [Holland 75]. Such algorithms are based upon the genetic/evolutionary metaphor where the solution is encoded in a chromosome that supports mutation, crossover and selection to allow the system to evolve.

2.4.1 Genetic Algorithms

This technique, introduced by John Holland [Holland 75], codifies a solution as a string that represents its genetic information. Although the original technique uses binary encodings, real valued encodings are as powerful and sometimes more natural to the problem [Antonisse 89].

Genetic algorithms differ from traditional search algorithms in the following four ways [Goldberg 89]:

- GAs usually work with a coding of the parameter set, not the parameters themselves. This is also known in the GA literature as indirect encoding. It means that a GA works with the instructions about how to obtain the values of the parameters, that is, with the original elements of a mapping whose images are the parameters. The mapping/instruction set is usually chosen to generate a search landscape easier for the GA than working directly with the parameters themselves.

- GAs search from a population of points, not a single point. This allows search to take place in parallel, making it easier to avoid the problem of local minima that is the weakness of hillclimbers and gradient descent methods. Moreover, the *building block hypothesis* [Goldberg 89] suggests that small chunks of good solutions (the particular chunks/parts whose presence makes that solution good) can be combined during crossover to generate even better solutions. Therefore,

a population of points provides a genetic pool where these chunks can appear independently to be combined at latter stages of the GA run.

- GAs use information on objective functions but not derivatives or other auxiliary knowledge. This makes GAs very useful for problems where no methods or heuristics about how to solve them are known or available. It is also useful for problems where the function to optimize is not derivable (like piece-wise or non-continuous functions). Just the final outcome of the objective functions is necessary to guide the search.

- GAs use probabilistic transition rules, not deterministic rules. Deterministic methods must ensure that progress is made to avoid getting stuck in cycles. GAs' probabilistic nature helps prevent such stagnation. Nevertheless, GAs can also get trapped in suboptimal solutions. It is a common practice to solve a problem by running a GA several times and selecting the best result. Even if such runs start from the same population the probabilistic nature of GA search will make the runs different and would, hopefully, be able to avoid such traps.

The GA consists of a population of chromosomes that encode solutions to a problem. Strictly speaking, an *evaluation* function measures the performance of the solutions, that is, how good a given solution is, while a *fitness* function determines reproduction opportunities. In most cases the fitness and evaluation functions are the same and the reproduction opportunities coincide with the measure of how good a solution is. The terms are treated as the same by many authors. In this work both terms will be interchangeable unless specifically stated otherwise.

2.4.1.1 The canonical GA

The canonical GA [Holland 75], after initialization where a population of (usually) random chromosomes is created, is executed in the following way:

- First, a selection process is undertaken. In this process solutions are selected using one of several criteria based on fitness such as described in [Ross 98, Whitley 01]. One solution can be chosen several times if it is fortunate and has a high fitness, thereby creating an intermediate generation.

- Second, depending on whether crossover is used, couples within such an intermediate generation are chosen according to a parent selection criterion and are then crossed to yield one or two new offspring. These new offspring go to the next generation or compete with their parents to enter in the new generation.

- Finally, those solutions that are going to be part of the new generation will undergo mutation with a given probability. A mutation is a random change within a chromosome. This is done to increase the diversity of the genetic information, due to the fact that after several generations the diversity of the chromosomes decreases and some chunks of the chromosome may become the same for all population members. Without mutation a chunk will remain with no changes and the information it contains will not evolve (and improve) further.

This process is repeated until an end criterion is reached. Such criteria include when a given number of evaluations or when all the members of the population are equal (or equal up to a certain degree). Note that the time performance of a GA is usually measured by the number of evaluations. An evaluation is a call to the evaluation or to the fitness function. The use of GAs involves several parameters that have to be tuned to obtain good performance, including the size of the population, the mutation rate, the parent-selection criteria, the replacement criteria, the type of crossover used and the type of mutation used. Most of those parameters are summarized in table 7.7 in further detail.

2.4.1.2 Advanced Genetic Algorithms

When applying GAs to highly complex application domains, important problems may arise. Possibly the most common is premature convergence. This happens when the population converges too early onto a non-optimal local minimum [Davis 91], and is unable to escape from it. Other problems may be caused by deceptive functions, which are, by definition, hard for most GAs to solve. Noisy functions [Goldberg *et al.* 92a, Goldberg *et al.* 92b, Lomborg 94] and the optimization of multiple criteria within GAs also can cause difficulties [Fonseca & Fleming 95]. In an attempt to overcome these problems, new, more advanced types of GA have being developed. Without pretending being a complete list these advanced genetic algorithms include, among others:

- Parallel GAs, where multiple processors are used in parallel [Mühlenbein *et al.* 91, Levine 94], exploiting certain parallelisation potentials that are inherent to the definition of the GA that is being run.

- Distributed GAs, where multiple populations are separately evolved with few interactions between them [Whitley & Starkweather 90, Mühlenbein 92] also known as the "island model". They evolve separate populations by exchanging, now and then, the best members of the populations. The idea is to promote diversity as each population will have their own evolution. If some island converges prematurely the new genetic information brought by the fittest of other populations would help put them back on track.

- GAs with niching and speciation, where the population within a GA is segregated into separate 'species' [Horn 93, Horn & Nafpliotis 93, Horn *et al.* 94a]. These techniques also promote diversity, but this time within the population. There are several ways to obtain such an effect, including:

  - Preselection [Cavicchio 70, Mahfoud 92]: It consists of replacing one of the parents. This way several family lines are expected to appear.

  - Sharing [Goldberg & Richardson 87, Deb 89, Deb & Goldberg 89, Horn 93, Mahfoud 93]: Here the fitness function is penalized on the basis of how similar a particular individual is to the rest of the population. This means that being different attracts very low penalty and, therefore, the individual is promoted. It is called sharing because a group of very similar individuals effectively share the reward that yields the place they point to in the search.
  - Crowding [De Jong 75, Mahfoud 92]: This method proposes that each new individual replaces a similar individual (that is worse, of course). It is expected, therefore, that diversity is not severely harmed as each individual can only remove a member of its own "species". In a way, preselection is a form of crowding, as it is expected that children are similar to their parents.
  - Boltzmann tournament selection [Goldberg 90, Mahfoud 91]: These have been reported to produce similar effects to niching.

- Messy GAs (mGAs), where a number of 'exotic' techniques such as variable-length chromosomes and a two-stage evolution process are used [Deb 91, Goldberg *et al.* 91]. The first stage tries to obtain promising building blocks while the second mainly rearranges them in an attempt to find optimal solutions.

- Multi-objective GAs (MOGAs) [Schaffer 85, Schaffer & Grefenstette 85, Goldberg 89, Srinivas & Deb 95, Bentley & Wakefield 96], where multiple objectives are allowed to be optimized within the evolution process. This embodies several areas of research, with the most common approaches including:

  - Aggregation approach [Goldberg 89]: It works by adding each objective reward function value into a single fitness value. Usually the reward values

of each independent objective are normalized to allow aggregation in an unbiased way. These methods allow some objectives to be regarded as more important than the others, by weighting the reward values in the aggregation. An interesting review of different aggregation methods can be found in [Bentley & Wakefield 96].

- Non-Pareto approach [Schaffer & Grefenstette 85]. It works by rotating the objectives to be taken into account for each generation. Being a generational GA, for each individual objective the selection method is repeated to fill up a portion of the mating pool. Then the entire mating pool is thoroughly shuffled to apply crossover and mutation operators. The algorithm tends to work efficiently for a certain number of generations. Unfortunately, it seems that in some cases it suffers from its bias towards some individuals or niches, ending up mostly with members that are individual objective champions.

- Pareto-Based approach [Goldberg 89]: This is based on the concept of the Pareto-optimal set, which is composed of a family of individuals that is optimal in the sense that no improvement can be achieved in any objective without degradation in others. The concept of Pareto-optimality is closely linked with the concept of dominance among individuals. The definition of dominance goes as follows: Assume that the multi-objective GA tries to optimize individuals $x \in X$ in a set of objective functions $f = (f_1, f_2, \ldots, f_q)$. All these individuals must belong to the feasible region $\mathcal{F} \in X$ (allowing the representation of constrains imposed over the possible solution values). Let x^0, x^1 and $x^2 \in \mathcal{F}$.

  * x^1 is said to be dominated by x^2, if $f(x^1)$ is partially less than $f(x^2)$,

that is:

$$\begin{gathered} f_i(x^1) \leq f_i(x^2),\ i \in P \\ f_j(x^1) < f_j(x^2),\ j \in Q \\ P \cap Q = \emptyset \\ P \cup Q = \{1, \ldots, q\} \end{gathered} \tag{2.66}$$

* x^0 is the *Pareto-optimal* (or *non-dominated*), if $\nexists x \in \mathcal{F}$ such as that x dominates x^0.

The algorithm then sorts individuals as dominated or non-dominated and gives more chances of reproduction to the non-dominated members of the population. As dominance is an expensive thing to check several methods to estimate dominance have been developed. Among the different examples it is worth mentioning the Pareto domination tournaments in the niched-Pareto GA (NPGA) found in [Horn *et al.* 94b] and a non-dominated sorting GA (NSGA) developed in [Srinivas & Deb 94].

These multi-objective methods usually use niching or other diversity promotion methods to spread the individuals into different areas of the Pareto-optimal set. This allows for different optional solutions, each of them representing a different trade-off of the diverse objectives. For an exhaustive collection of papers in multi-objective GAs field see [Coello 03].

- Hybrid GAs (hGAs), where GAs are combined with other search or representation techniques [Radcliffe & Surry 94, Poloni 95, George 96]. There are many examples of such combinations. In particular, the so-called memetic algorithms [Moscato 89, Radcliffe & Surry 94, Burke *et al.* 95] perform, between generations, certain cycles of hill climber algorithms for every member of the population. GAs may also be combined with stochastic methods such as simulated annealing [Ho & Lee 00]. Examples of combination of other evolutionary techniques are also present, including

GA/GP hybridizations as reported in [Howard & D'Angelo 95, Lee *et al.* 96]. Other hybridizations combine GAs with some heuristics defined in particular problems [Fang *et al.* 94, Reeves 96]. Of course, hybrids of GAs and fuzzy systems have also been developed [Fukuda & Shibata 94, Joo *et al.* 97, Gómez Skarmeta & Jimenez 99, Setnes & Roubos 00, Cordón *et al.* 01]. Likewise, neural networks are many times mixed with GAs [Miller *et al.* 89, Kitano 90, Schaffer *et al.* 92]. Even the latest optimization techniques such as ant colonies [Dorigo *et al.* 96, Bonabeau *et al.* 99] can not escape being hybridized with GAs [Acan 02].

- Structured GAs (sGAs), where parts of chromosomes are allowed to be switched on and off using evolveable control genes [Dasgupta & McGregor 92, Parmee 99, Gómez Skarmeta & Jimenez 99, Gómez Skarmeta *et al.* 99]. For example, in [Gómez Skarmeta & Jimenez 99] the chromosome represents a set of fuzzy rules plus a control part that decide if that particular rule is going to be activated or not.

2.4.2 Other Evolutionary Techniques

Evolutionary computation [Heitkötter & Beasley 97, Bäck & Schwefel 93] is a very active and alive field. Genetic algorithms have become just a line of research in a broader evolutionary inspired field. As indicated above, alongside with GAs stand Evolutionary Strategies, Genetic Programming and Evolutionary Programming. They share the concepts of population based search and the use of evolution as the searching engine. It is far beyond this background chapter to cover all of them, but in the sake of completeness the main related lines are mentioned below with some references.

2.4.2.1 Evolutionary Strategies

They were first developed in Germany in the mid-sixties [Schwefel 65]. In evolutionary strategies the search points are n-dimensional real vectors. The fitness value is the objective function itself. The move operator is mainly mutation although recombination (shuffling) of elements is also present. Alongside with the vectors the variances and covariances of the vector components are also stored and used to self-adapt the step size of the mutation. The population replacement follows two main approaches, namely $(\mu+\lambda)$-ES or (μ,λ)-ES with μ being the number of members of the population and λ the number of children in a given generation. $(\mu+\lambda)$-ES selects, for the new population, μ elements among the combination of μ and λ elements. (μ,λ)-ES only takes λ members into consideration for the new population (that will be of size μ). For a broader explanation of Evolutionary Strategies see [Schwefel 81, Bäck & Schwefel 93].

2.4.2.2 Evolutionary Programming

Also developed in the sixties [Fogel *et al.* 66], evolutionary programming was designed to work on the state transition tables of finite state machines (thereby being a form of programming). This approach is rather similar to Evolutionary strategies. The only differences resides in that mutation is the only move operator available and that the new population is obtained as the union of the best half of parents and the best half of children (a kind of $(\mu+\lambda)$-ES). More information on Evolutionary Programming can be found in [Fogel 92].

2.4.2.3 Genetic Programming

Genetic Programming (GP) was developed in the early nineties [Koza 90, Koza 92]. GP is a real form of automated programming. Although it was not the first to hint in this direction (evolutionary programming, for example, is another) it definitely sets

the changes in evolutionary computing that creates a branch of its own. GP evolves LISP style trees [Koza 92] where the nodes are operators and the leafs are constants or variables (terminals). There exist several operators to modify the tree, including those to change a subtree for a leaf, or a leaf for a subtree, to interchange subtrees among parents, to rename a subtree as a new terminal (effectively creating subroutines), etc. The final tree is treated as a LISP program to solve the particular problem at hand. The syntax of the language is ensured by the way the operators act. The semantics are changed so all syntactically correct programs are semantically correct (for example, if a division by zero is encountered a default result of one is returned; infinite loops are avoided by setting a maximum number of loops, etc). Many interesting extensions have also been added to deal with, for example, the use of a memory. The result is a very active branch with some astonishing results [Koza *et al.* 99]. More information can on Genetic Programming be found in [Koza 98].

2.5 Summary

This chapter has covered two main sections, one about fuzzy theory and another about evolutionary computing. The former gave a review of the basic concepts of fuzzy logic, fuzzy sets, and fuzzy inference. In addition, several types of fuzzy set, not just the ones to be used, were defined for completeness. The two fuzzy approaches, approximative and descriptive were also detailed. The latter section served as an introduction to evolutionary computation in general and genetic algorithms in particular. GAs are used to implement the present work in the subsequent development of this thesis.

Chapter 3

Automatic Generation of Fuzzy Rules

> " Never accept the proposition that, just because a solution satisfies a problem, that it must be the only solution."
>
> *Raymond E. Feist*

3.1 Introduction

Fuzzy Rule Based Systems can be divided into two main categories: Approximative and Descriptive. The former is mainly concerned with accuracy and flexibility. The latter is focused on interpretability.

Approximative Fuzzy Systems have many options available. They can use any kind of fuzzy set in the antecedent. Each rule can have its own membership description for each variable. It can use any T-Norm and T-Conorm pair (see sections 2.3.3.1, and 2.3.3.2) in its inference. The consequent can be anything: a singleton, a fuzzy set or an expression. Basically, everything is allowed to be tuned, modified or used in order to attain the maximum efficiency and accuracy. However, such systems are very difficult to interpret as the sets end up as rather obscure mathematical expressions. Even graphical representations may not help much because free tuning may lead to sets that represent concepts that are difficult to grasp. Approximative sets are generated to

fit the data and not the human mind that reads the sets, a human mind that usually has its own way to categorize (partition) a given domain. Besides, as the very reasoning procedure can also be tuned, it may end up in a mathematically sound but unintuitive way of doing it. This kind of fuzzy system is usually related with, but not restricted to, the TSK rules [Takagi & Sugeno 85], although TSK systems can be created to be descriptive.

Descriptive Fuzzy Systems, on the other hand, suffer from lots of restrictions and are rather inflexible. The domains are partitioned beforehand (usually by the humans that are developing or going to use the system, therefore enforcing human-like categorization) and the fuzzy sets are labelled. Those partitions should follow certain guidelines and restrictions to become descriptive [Valente deOliveira 99]. Once the sets are defined the rules are only allowed to use such sets. This effectively defines a finite set of possible rules. In fact, assuming a MISO (Multi-Input Single-Output) fuzzy rule system of *AND* connected rules, if there are M different input variables and each variable has $L_i, i = 1..M$ different sets (i.e. labels) including a $WHOLE$ set (so rules can omit that particular variable) and O different possible outputs then the number of possible rules are:

$$L_1 \times L_2 \times ... \times L_M \times O$$

It is easy to see this increases with the addition of new variables and for new labels, the so-called "curse of dimensionality" [Ishibuchi *et al.* 97, Dick *et al.* 99]. The higher the number of variables available, and the higher the cardinality of the fuzzy partition for each variable, the higher (exponentially) the number of possible rules. This is of course generally true for approximative models as well.

The main problem related with descriptive systems is granularity due to the fact that the fixed sets produce a fixed grid that may not fit well the data. Improvements have been suggested to overcome this problem. One particular approach is to allow modifications in the definition of the fuzzy sets, but this soon makes the descriptive

sets degenerate into approximative ones as usually not many restrictions are imposed. Another one is to use linguistic hedges [Zadeh 75] in the fuzzy rules. This gives a new degree of freedom but at the cost of increasing the possible number of rules and rendering some descriptive fuzzy rule generation methods useless. Descriptive fuzzy systems have been usually associated with Mamdani [Mamdani & Assilian 75] rules (although Mamdani rules can be approximative as well).

3.2 Rule Generation

The first fuzzy rule bases were generated by human experts although very soon methods to obtain the rules automatically started to develop. Early rule based systems were descriptive as the introduction of fuzzy sets was intended to allow computing with words, the ultimate goal of fuzzy logic [Zadeh 96], and therefore the interpretability was the most important objective. As data was replacing first hand expertise the methods to generate fuzzy rule algorithms became more and more approximative as each new improvement turns the knot more and more in accuracy at the cost, usually, of the interpretability. Also, there was a kind of marriage between neural networks and fuzzy rule-based systems. Even their equivalence under certain restrictions was proven [Jang *et al.* 93]. However, although the neurofuzzy systems can produce rules, the rules are approximative and the interpretability is not ensured.

In this review, unless stated otherwise it is assumed that the method for generation will look for scattered fuzzy rule systems (that is, it does not ensure complete coverage) with the smallest number of rules possible. Also, it is assumed that the generation is data driven and not under direct human supervision, though the training samples provided are typically consistent with human understanding (in terms of input-output patterns). In addition, for reinforcement learning [Samuel 59] a measurement of the performance must be available.

Algorithms for producing fuzzy trees also allow the generation of fuzzy models

but they are not considered here in detail. Recently, decision trees, originally introduced in [Quinlan 86] have gained popularity. The main algorithms were the ID3 [Quinlan 86], the CART [Breiman *et al.* 84] and the C4.5 [Quinlan 93]. Very soon they were fuzzified like Fuzzy-ID3 [Umanol *et al.* 94], Fuzzy-CART [Jang 94] and many others [Ichihashi *et al.* 96, Janikow 98, Wang & Hong 98]. Unfortunately, the power of Fuzzy Decision trees is limited as such trees have been reported to have poor learning accuracy [Wang *et al.* 00, Yuan & Shaw 95], although some improvements have been proposed as in [Tsang *et al.* 00]. For more complete information regarding decision tree generation refer to [Janikow 98, Wang *et al.* 01]

Most of the existing work makes use of linear piece-wise fuzzy sets, for simplicity and easiness of computation [Shen & Leitch 93]. However, sigmoidals are also widely used, especially when neural-networks are around, as the back-propagation [Rumelhart *et al.* 86a, Rumelhart *et al.* 86b] algorithm needs differentiable functions. There are no extreme benefits of one versus another although sigmoidals provide full coverage as the membership function is only in the infinity values.

Researchers have also reported improvements in the performance of fuzzy models when ellipsoidal rules are used [Dickerson & Kosko 96, Abe & Thawonmas 97, Abe 98, Abe 99]. The ellipsoidal rules evolved from using radial basis functions as fuzzy set definitions and are closely related with neural networks. In [Dickerson & Kosko 96], for example, neural networks are used to generate and to tune such ellipsoids. The projection of the ellipsoids yield triangular (although they may have other shapes) fuzzy sets for each variable, with their combination forming the fuzzy rule. This is commonly applied to function approximation. In [Abe & Thawonmas 97, Abe 98, Abe 99], however, the ellipsoid shaped fuzzy sets are derived from the kind of clustering used to generate the rules (the clusters used have an ellipsoid shape).

3.2.1 Generation of Descriptive Rules

In pure descriptive generation systems there are various ways of obtaining rules, including:

- Considering all possible rules one by one and selecting those that fit data. The output of the rule is obtained selecting the dominant class [Lozowski *et al.* 96] for the data that is covered by the rule.
- Addressing each data point, obtaining the rules that best cover that datum and, in case any other data point is covered by the same rule, solving any clash of different outputs by maximizing certain output criteria or by prioritising some data versus another [Wang & Mendel 92, Ishibuchi *et al.* 99].
- Applying a search technique over the space of the possible rules, like an evolutionary algorithm [Cordón & Herrera 99, Jin *et al.* 99, Akbarzadeh-T *et al.* 98], or any other global search method.
- Following a fuzzy decision tree generation technique such as Fuzzy-ID3 [Umanol *et al.* 94].

3.2.2 Generation of Approximative Rules

Many methods have been developed for approximative rules, with a great majority of them being small improvements over previous ones. Traditionally the process of modeling has been divided into structure and parameter identification [Wang 92, Yager & Filev 94, Kosko 97, Jang *et al.* 97]; the former chooses the number and shape of the rules and the latter the parameters of the sets used in the rules. Nevertheless, in many cases, the tuning changes both structure and parameters. It is easier to divide the process into generation and tuning steps. The first explains the way that an initial but crude rule base is obtained and the second shows the way that such an initial rule base is improved until the resulting model is acceptable. Usually, rule generation and tuning

are not entangled although sometimes the tuning is done by repeating the generation process with a smaller problem in a recursive way.

The majority of automatic rule generation for approximative rules follows one of the following methods:

- Transform methods: A descriptive fuzzy rule-based system is deliberately corrupted and degenerated into an approximative system, by modifying the membership definitions of the fuzzy sets of the rules.

- Partition methods: This group follows the strategy of applying an iterative partition to the data space when a certain measure of the error is unacceptable or the number of resulting rules is within a given limit. The approach starts with fuzzy sets that cover the entire input space and divides them in places where it is expected that the error can be reduced most. Each time a set is divided the number of rules is doubled and the rules' outputs recalculated [Chen *et al.* 98]. Other partition methods [Kim 97] perform a variable fuzzy partition first and then extract rules using the partition.

- Neural networks: A neural network can be used to obtain a model and the rules are extracted from the network. The neural network is usually arranged in a way that facilitates the rule extraction. The famous ANFIS (Adaptative Neuro-Fuzzy Inference System) [Jang *et al.* 97] arranges a fuzzy rule base as a neural network so that back-propagation [Rumelhart *et al.* 86a, Rumelhart *et al.* 86b] (and indeed any other neural network optimization technique) can be applied to tune the antecedent part.

- Clustering methods: This type of rule generation approach works by applying a clustering technique [Bezdek & Pal 92] to the data available and then covering such generated clusters with rules (one rule per cluster). Such a method projects the clusters in each dimension of the data set and generates fuzzy sets that

cover the coordinates of the training examples belonging to a certain cluster. A resultant rule is a conjunctive expression of all the sets generated for each underlying input variable. The output of a rule is obtained as weighted average of the outputs of the data covered (or the most common class). This method is one of the most popular.

- Random methods: This is just to mention that a random start is as valid as any other if an optimization process is going to be used later. Nevertheless, many optimization techniques usually work faster starting from an already decent solution. In particular, genetic algorithms, for which randomness is essential to avoid premature convergence and stagnation, work better if a good start is identified [Fogarty 96, Kallel & Schoenauer 97] as mutation is there to provide such randomness.

3.3 Tuning Step

Tuning a fuzzy rule based system is basically done by one way only, that is search. Almost all the search techniques available have been applied. Among the search techniques used the most popular have been hill climbers [Sugeno & Yasukawa 93, Abe & Thawonmas 97, Abe 98, Abe 99], gradient descent [Nomura *et al.* 92, Marín-Blázquez *et al.* 99], genetic/evolutionary algorithms [Gómez Skarmeta & Jimenez 97, Jin *et al.* 99, Cordón & Herrera 99, Ishibuchi *et al.* 99], neural networks [Hayashi *et al.* 92, Dickerson & Kosko 96, Simpson 92] and reinforcement learning [Berenji & Khedkar 92, Berenji 96, Jouffe 98, Kim *et al.* 99]. They are driven only by error measures and usually modify the membership functions as well. In so doing, they generate approximate rules even if the original rules and membership functions were descriptive.

Of course, some of the above methods can be combined or serialized in what is called hybrid systems [Delgado *et al.* 98a, Gómez Skarmeta & Jimenez 99].

Although claims exist such that after tuning the resultant systems are descriptive, practically all of them do not fully ensure a model that can be readilyunderstood by human users. At most, they may be termed as pseudo-descriptive, even though some constraints can be imposed [Jin *et al.* 98], such that the generated membership functions may be linguistically approximated [Sugeno & Yasukawa 93] or even directly labelled [Setnes *et al.* 98b].

A particularly interesting method is to regain interpretability by aggregating final approximative sets until only a few remain for each variable [Setnes *et al.* 98b, Setnes *et al.* 98a, Setnes & Roubos 00, Roubos & Setnes 01]. The resultant sets can have interpretable properties [Valente deOliveira 99] for certain problem cases. Unfortunately, the labelling is done *a posteriori* and the resulting fuzzy sets may not be labelable at all as the Iris Flower problem [Fisher 36] (sepal length) example given in [Setnes & Roubos 00] demonstrates.

3.4 Example Approaches

In this section, selected representative approaches for fuzzy rule generation are discussed. The following two sections will provide a more detailed account of two particular approaches that are used in the present work.

In [Jin *et al.* 99] Mamdani type fuzzy rules are obtained by an evolutionary algorithm which tunes the ruleset. The initialization is random but allows expert knowledge to be added. The tuning algorithm enforces coverage and distinguishability. The fuzzy sets employed are assumed to be Gaussian. The algorithm also coevolves the T-Norm used in the inference and in defuzzification. No control of the number of rules is enforced, however.

In [Cordón & Herrera 99] evolutionary processes are used in the generation and also in the tuning of a TSK system. This work starts by fixing a number of sets per variable and searching in the whole product space for rules that cover the data. A rule

is selected if its output class is deemed to be dominant. Then an evolutionary algorithm is used to decide the consequents of the rules. Once this generation method finishes the obtained rules are used to create the population of a new evolutionary algorithm in a Pittsburgh style (that is, each member of the population encodes a whole fuzzy rule system) [Smith 80] . This time the fuzzy memberships and the consequents are modified at the same time. The number of the resultant rules is the number of rules generated in the evolutionary generation method, so no control is imposed over this number.

In [Ishibuchi *et al.* 99] a fuzzy classifier induction mechanism is proposed. The generation is done through a process similar to the one in [Wang & Mendel 92]. The tuning is done by a genetic algorithm. The fitness is based on the number of the correctly classified data. It follows a Michigan style (namely, each member of the population encodes a rule, the whole population is the fuzzy rule system) [Booker *et al.* 89] , so the population size is the number of resulting rules.

Work also exists in proposing the building of fuzzy classifiers [Abe & Thawonmas 97, Abe 98] and function approximators [Abe 99] with ellipsoidal rules. The generation is done by clustering, and the tuning is done by hill climbing.

In [Emami *et al.* 98] a systematic methodology for fuzzy modeling is explained. It covers all aspects of developing a fuzzy system starting from attribute reduction up to inference-parameter adjustment. The generation of rules is carried out by clustering. The tuning is implemented by using the algorithm given in [Sugeno & Yasukawa 93], basically a hill climber. It also optimizes the reasoning mechanism by applying a parametrized reasoning mechanism (assuming crisp inputs). The parameters relate to the behavior of the T-Norms and T-Conorms and to the defuzzification method employed.

A piece of very interesting work can be found in [Velasco 98]. It describes an approach that uses random generation, and a Michigan style [Booker *et al.* 89] genetic algorithm as the tuning technique. This algorithm is used on-line, with the fitness of

the rules obtained by reinforcement learning [Hecht-Nielsen 90]. Of particular interest is the use of the so-called "Limbo" storage: a place to store the newly generated rules until proven significantly good to become part of the knowledge base. This helps avoid the degradation of performance when new genetic material is introduced due to its on-line operationality.

A descriptive but rather limited algorithm is presented in [Au & Chan 98]. This algorithm generates rules by exhaustive search but only among pairs of attributes. In fact, it extracts rules that relate a pair of attributes (IF x_i is L_{pi} THEN x_j is L_{qj}, with $i, j \in \{1 \ldots M\}$ and p,q within $|L_i|$,$|L_j|$, the number of possible labels for each variable, respectively). A selection among all possible pairwise combinations is then carried out through statistic measurements.

In [Jin *et al.* 98] another genetic algorithm generation and tuning approach is proposed. The fitness this time takes into account the inconsistency and incompleteness of the resulting fuzzy partition (as the membership functions are also tuned). The rules are selected using a binary matrix of all possible candidates and evolving their inclusion or exclusion in the final fuzzy rule base. After tuning the ruleset, the resultant fuzzy rule-based system is converted into a radial basis function neural network. Afterwards a second tuning is performed over the neural network. Finally, the radial basis function is reverted back into fuzzy rules.

The work reported in [Mitra & Pal 96] suggests a fuzzy classifier that utilizes a fuzzy-extended Kohonen's self-organizing neural network [Kohonen 89]. The connection weights of the neural net form the knowledge base. Rules can be extracted when an example is presented to be classified, that is, when a justification in form of IF-THEN rule is obtained. As in Kohonen self-organizers, the initialization is done randomly and the "tuning" is implemented by stabilizing the weights at the presence of the training examples. The presence or absence of a variable in the fuzzy rule for justification is determined by its impact upon the activation of the corresponding neuron. The fuzzy sets to be used in the justification rules are predefined and, in

this sense, the resultant rules are pure descriptive. The input to the net can be crisp values, fuzzy sets or a combination of linguistic hedges and fuzzy sets. To infer, the net iteratively asks for variable values in an order of usefulness given the previous values of the variables of a particular example to be classified. If there are missing values such variable may have been overlooked. The certainty of the inference depends on the difference between the membership value of the winner class with the runner up class. Although this method can be considered descriptive, and justifications in descriptive form can be obtained, as the knowledge base is in a network, it is difficult to investigate how the whole underlying system behaves.

In [Mitra *et al.* 97] a fuzzy multilayer perceptron is adopted to be the knowledge base, but this time the construction of a fuzzy rule is suggested as the final step. This is proposed to be done by either providing the system with an exhaustive combination of possible antecedents (as the inputs can be fuzzy sets), leaving the system to generate the output, or presenting a number of representative examples with known outputs and generating the antecedents via backtracking.

Regarding reinforcement learning in [Jouffe 98] the generation of the antecedent of the rules is done by humans. The consequent is tuned by Fuzzy Actor-Critic Learning or by Fuzzy Q-Learning (based on AHC [Barto *et al.* 83] and Q-Learning [Watkins 89] respectively). In [Berenji & Khedkar 92] the rules are arranged in a neuro-fuzzy way and the reinforcement learning modifies also the membership functions of the antecedent parts, although the learning can be applied just to consequents if desired.

In [Tunstel *et al.* 96, Akbarzadeh-T *et al.* 98], within several techniques discussed is a genetic programming method used to generate a decision tree (and therefore a set of rules). The fuzzy memberships are fixed and hence the approach can be regarded to be a descriptive method. In this work, a controller for a robot is developed to illustrate the ideas. The genetic programming searches through a subset of rules that produce a better behavior of the robot. This approach is faster than performing brute search over the whole space but the efficiency gain over other search techniques seems dubious.

Different techniques for generating and tuning classifiers can be hybridized. For example, in [Gómez Skarmeta *et al.* 01] a proposal is given to generate and tune N-dimensional fuzzy rules where the antecedent is a fuzzy set defined in the whole input space. This helps avoid the problem of inverse projection (see the corners of the rectangle in Fig 3.1 where points not covered in the original multi-dimensional fuzzy set become covered due to spurious firing when the projections are combined in an AND rule). The method also uses Differential Evolution [Storn & Price 97] and genetic algorithms to fine tune the rules. The work actually integrates attribute reduction, a substractive fuzzy clustering [Chiu 94] and ANFIS [Jang *et al.* 97] within a common framework.

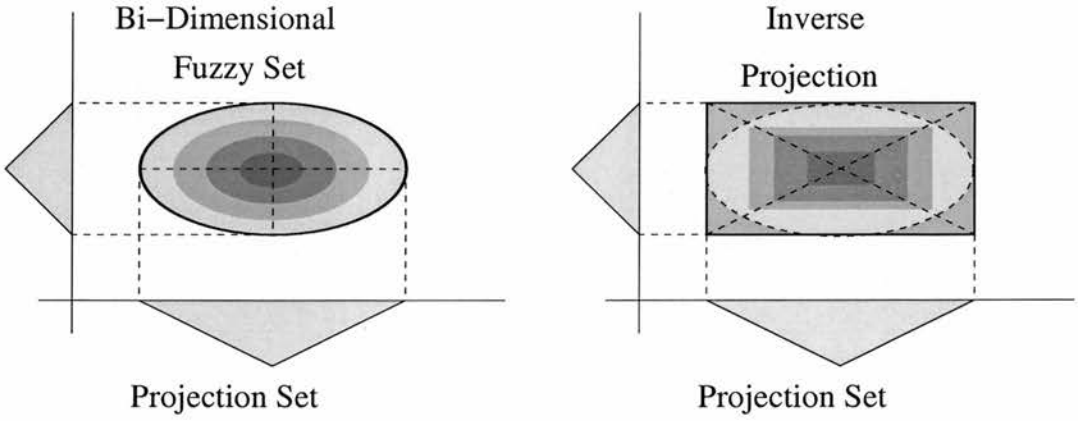


Figure 3.1: The problem of inverse projection

Finally, in [Marín-Blázquez *et al.* 99] a proposal for a community of software agents cooperating to generate fuzzy models is outlined. Several rule generation techniques (various clusterings and an uniform distribution) and different tuning methods (gradient descendant and Least Vector Quantifier) compete to be chosen as the definitive generation/tuning method that will ultimately be applied. This full approximative approach for function approximation includes background rules (multi-dimensional fuzzy rules that activate when a point falls outside the currently covered region and that can be tuned also) to ensure coverage and multiple error layers.

Hence, an error layer is an extra fuzzy model that approximates the error committed by the existing model. Its output is added to the model to compensate the error. As soon as the existing model is fully tuned an error layer fuzzy model can be generated and tuned in a distributed environment and when the error layer model is finished it is incorporated into the existing one. As inference is parallel the only extra overhead is the addition of the results. The layers can recursively increase until no further improvement is possible or there are no more resources available.

3.5 Lozowski's Algorithm

In this research, to obtain a fuzzy ruleset, Lozowski's pure descriptive induction algorithm [Lozowski *et al.* 96] is used. This algorithm works by exhaustive search making it difficult and impractical for high dimensional problems. Due to the nature of exhaustive search, however, the results obtained are the best possible for a descriptive model that uses no hedges. For scaled-up applications of the ideas given here an alternative descriptive modeling method may be necessary.

Lozowski's algorithm generates a hyperplane of candidate fuzzy rules (see Equation 2.60) by fuzzifying the entire dataset using all permutations of the inputs. Thus, a system with M inputs, each of which has a domain fuzzified by f_j fuzzy sets ($1 \leq j \leq M$), the hyperplane is fuzzified into $\prod_{j=1}^{M} f_j$ M-dimensional hyperboxes, each representing one vector of rule preconditions. Each hyperbox $\underline{p} = \langle D1, D2, \ldots, D^M \rangle$ may lead to a fuzzy rule, provided that training examples support it.

To obtain a measure of what classification applies to a hyperbox, fuzzy min-max composition is used, although any other pair of T-Norm and S-Norm would do. The input pattern of each example is fuzzified according to the fuzzy sets $\{\mu_{D1}, \mu_{D2}, \ldots, \mu_{D^M}\}$ that make up hyperbox $\underline{p}$. For each example $\underline{x} = \langle x_1, x_2, \ldots, x_M \rangle$, the T-Norm of it with respect to hyperbox $\underline{p}$ and classification c is calculated as

follows:

$$T_c^{\underline{p}}\underline{x} = \min\left(\mu_{D1}(x_1), \mu_{D2}(x_2), \ldots, \mu_{DM}(x_M)\right) \tag{3.1}$$

To give a measure of the applicability of a classification to hyperbox $\underline{p}$, the maximum of all T-Norms with respect to $\underline{p}$ and c is then calculated and this is dubbed an S-Norm:

$$S_c^{\underline{p}} = \max\left\{T_c^{\underline{p}}\underline{x} \mid \underline{x} \in C_c\right\} \tag{3.2}$$

where C_c is the set of all examples that can be classified as c. This is iterated over all possible classifications to provide a full indication of how well each hyperbox applies to each classification.

A hyperbox generates at most one rule. The rule's preconditions are the hyperbox's M co-ordinate fuzzy sets. The conclusion is the classification attached to the hyperbox. Since there may be S-Norms for more than one classification potential contradictions may exist over a given classification. Such contradictions are resolved by using the *uncertainty margin*, ε ($0 \leq \varepsilon < 1$). An S-Norm assigns its classification on its hyperbox if and only if it is greater by at least ε than all other S-Norms for that hyperbox. If this is not the case, the hyperbox is considered undecidable and no rule is generated. The uncertainty margin introduces a trade-off to the rule generation process between the size and the accuracy of the resulting ruleset. In general, the higher ε is, the fewer rules are generated, but classification error may increase. A block diagram of this algorithm is shown in figure 3.2. A fuller treatment of Lozowski's algorithm in use for descriptive modeling can be found in [Lozowski *et al.* 96, Shen & Chouchoulas 00].

3.6 ANFIS: A Neuro-Fuzzy Inference System

This adaptive neural network based system obtains a model consisting of approximative fuzzy rules. It simulates the TSK (Takagi-Sugeno-Kang) reasoning mechanism [Sugeno & Takagi 83, Takagi & Sugeno 85], and it shares its adaptive

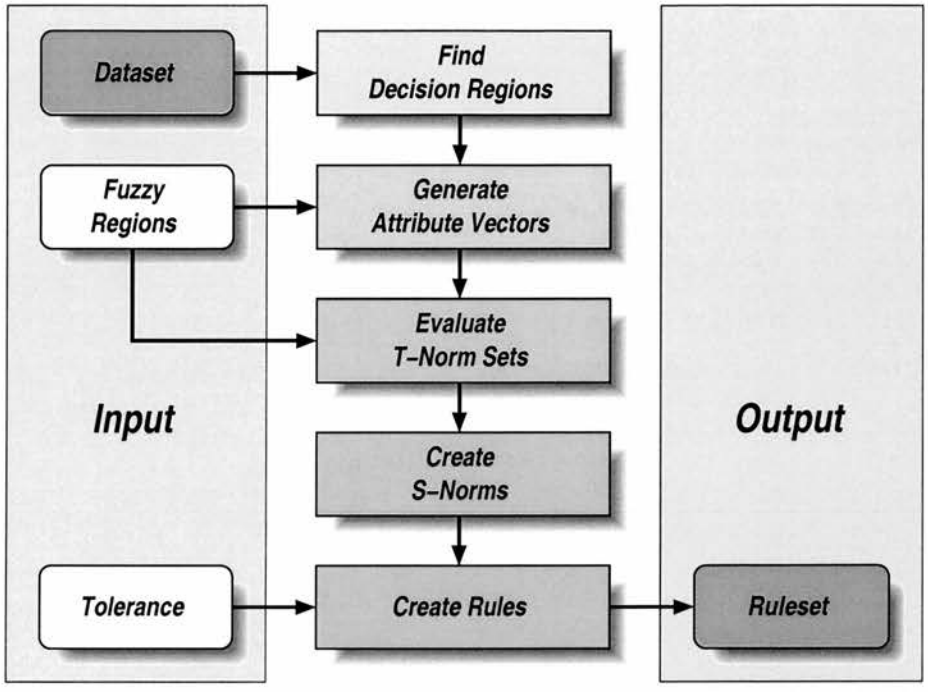


Figure 3.2: Lozowski's algorithm

capabilities to tune the parameters of the rules, which may have been estimated using some previous clustering algorithm. The output of clustering algorithms is a set of centroids $\{c_1, \ldots, c_t\}$, with each element describing training set points with similar features. These clusters become the antecedent fuzzy sets, which in TSK form are:

$$R_i : \text{IF } x_1 \text{ is } A_{1i} \text{ AND } \ldots \text{ AND } x_n \text{ is } A_{ni}q \text{ THEN } y_i = p_{0i} + p_{1i} \cdot x_1 + \cdots + p_{ni} \cdot x_n \tag{3.3}$$

where $x_1, .., x_n \in \mathbb{R}$, i.e., they are crisp values. Fuzzy sets $A_{1i}, \ldots, A_{ni}$ are obtained as a convex closure of centroid c_i, that is, projections on each axis of the centroid membership. Each consequent is a linear function with crisp inputs $x_1, .., x_n \in \mathbb{R}$, crisp output $y_i \in \mathbb{R}$, and crisp parameters $p_{0i}, \ldots, p_{ni} \in \mathbb{R}$ (it can be observed that if $p_{1i}, \ldots, p_{ni}$ are 0 then the consequent is a crisp constant p_{0i}). Using approximated

reasoning and the centre of area aggregation mechanism, the inferred output for a TSK model with t rules is obtained using equation (3.4):

$$f = \frac{\sum_{k=1}^{t} w_k \cdot (p_{0t} + p_{1t} \cdot x_1 + \cdots + p_{nt} \cdot x_n)}{\sum_{k=1}^{t} w_k} = \sum_{k=1}^{t} \bar{w}_k \cdot (p_{0t} + p_{1t} \cdot x_1 + \cdots + p_{nt} \cdot x_n) \tag{3.4}$$

where the fire strength for each rule is:

$$w_k = \prod_{j=1}^{n_t} \mu_{A_{jk}}(x_j) \tag{3.5}$$

that can be normalized:

$$\bar{w}_k = \frac{w_k}{\sum_{l=1}^{t} w_l} \tag{3.6}$$

Figure 3.3 shows the ANFIS architecture for two rules with three sets for each simple variable in the antecedent. It is composed of five layers. Nodes located in the same layer performs a similar task. The nodes of first and fourth layers are adaptive nodes, that is, they will be modified on learning. The values of these adaptive nodes are the parameters of the antecedents and consequents respectively.

It is interesting to note that the ANFIS architecture is a smart layout of fuzzy rules to act as a neural network. Learning is achieved combining back-propagation and least-squares methods. In each cycle the learning method runs a forward and a backward step. In the forward step, for each input vector, the net is evaluated until the fourth layer. Then, the parameters of the consequent are estimated using a least-squares estimator. After that, errors are calculated from each pair of network output and desired output. Finally in the backward step, the errors are propagated and the parameters of the antecedents are modified (back-propagation). More details about the generic ANFIS can be found in [Jang *et al.* 97].

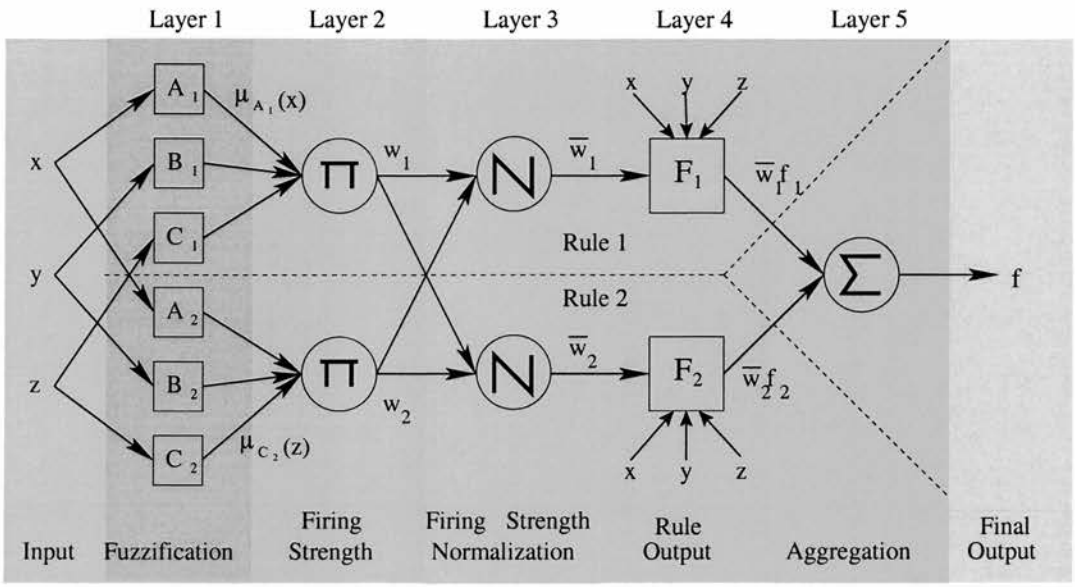


Figure 3.3: ANFIS network structure

Note that the general version of ANFIS has a real valued output and that it itself is also an additive system (the output is aggregated as the collection of the different evidences yielded by all fired rules). For this particular work a modified version of ANFIS was created. The modified ANFIS outputs integer values (that encoded the different classes). It also fires only one of the rules (as it is used as a classifier). This is achieved in layer 3 by modifying the $\bar{w}_k$ so that only one of them has a value of 1 and the rest are equal to zero:

$$\bar{w}_i = 1, \ \forall j, j \neq i, \ \bar{w}_j = 0, \quad i, j \in \{1, \ldots, t\} \mid w_i = \max_{k=1}^{t} w_k \tag{3.7}$$

3.7 Summary

In this chapter most of the current techniques for automatic generation of fuzzy rules have been reviewed. Distinct approaches to fuzzy model generation were addressed, with a focus on the discussion of approximative and descriptive fuzzy modelling

mechanisms. In particular, two specific algorithms that will be used for experimental studies in the thesis were introduced.

Chapter 4

Hedges

"Like a tree in the Autumn wind, it is far better to bend than to break."

Aesop

4.1 Introduction

Linguistic hedges were introduced by Zadeh in [Zadeh 75]. A linguistic hedge modifies the shape of a fuzzy set's definition, causing changes in the membership function [Cox 94]. That is, a hedge transforms one fuzzy set into another. The meaning of the transformed set can be extracted from that of the original set and that embedded in the hedge applied. There are several kinds of hedge depending on the type of transformation they perform on the membership functions. Examples of hedges include VERY, EXTREMELY, ABOVE, BELOW, etc. The order in which they are applied is important because NOT VERY Big has a different meaning to VERY NOT Big.

The definition of hedges has more to do with common sense knowledge in a domain than with mathematical theories. The following work reflects this observation. Although the same linguistic label may be used to express different transformations in the literature the corresponding type of these transformations

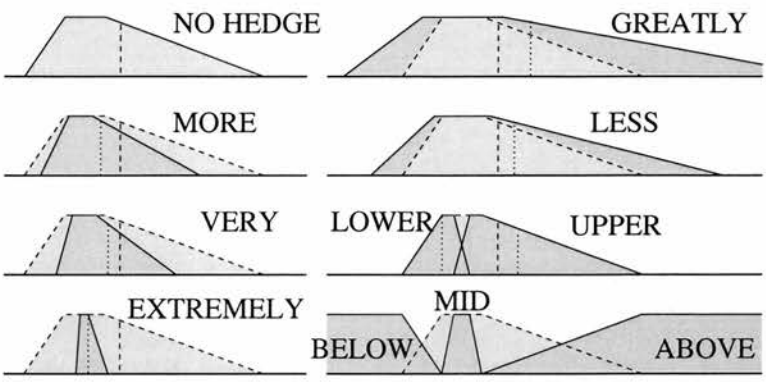


Figure 4.1: Hedges applied to an irregular trapezoid and how the way they change the center ofn understandable gravity

remains the same, varying only in terms of detailed parameter settings. For example, dilation/intensification hedges are often implemented by applying powers to the original set membership values. That is, given the original membership function $\mu_A(x)$ of a fuzzy set A and hedge H, the membership function of $H \cdot A$ is $\mu_{H \cdot A}(x) = \mu_A^e(x)$, where the exponent e is greater than 1 for intensification and less than 1 for dilation. Different values can be assigned to the exponent e; for hedge EXTREMELY for instance, $e = 2$ is used in [Cox 94], while $e = 8$ is employed in [Jang *et al.* 97].

However, conventional definitions of the hedges such as those examples shown above, do not result in significant changes on trapezoid fuzzy sets, which are most commonly used for computational simplicity. In particular, the full membership part of a trapezoid membership function does not change at all. In this thesis a different set of hedges is considered, which may be applied to dilute or intensify the original fuzzy sets by shrinking or expanding any parts of the trapezoids. In addition, three new hedges called UPPER, LOWER and MID that do not appear in the literature are also proposed.

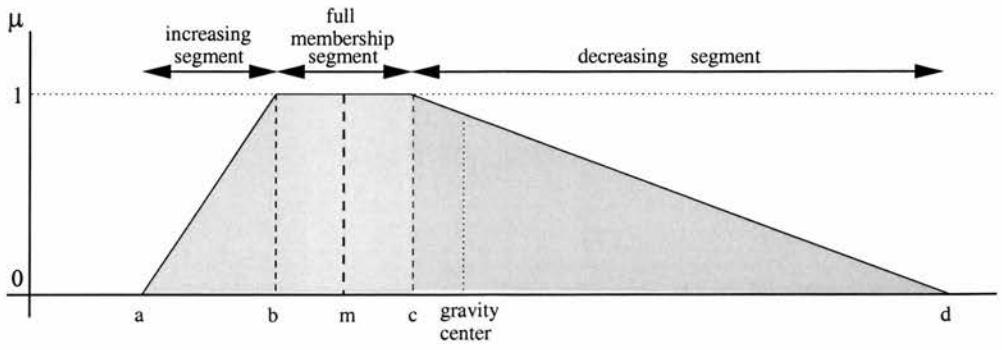


Figure 4.2: A trapezoidal membership function

4.2 Revised Hedges

A trapezoidal membership function, characterised by 4 parameters (a, b, c, d), consists of three consecutive segments as illustrated in figure 4.3. The application of dilation/concentration hedges should increase/decrease the size of these segments and, therefore, be implemented with a modification kept in proportion to the center of the full membership segment (m). This means that the center of gravity of the original fuzzy set and that of the modified will be different. In so doing, one set is always included in the other due to the piecewise linearity of the set membership definition. For concentration the modified set is included in the original and for dilation the original set is included in the modified. This makes more sense than if the shrinking/expansion is made with regard to the center of gravity (since it would otherwise produce shapes that may break the intuition of using the dilation/concentration hedges). For example, figure 4.4(i) shows the case where the hedge GREATLY is applied such that the center of gravity is preserved. This breaks the dilation principle over the dark green region where $\mu_{GREATLY \cdot S}(x) > \mu_S(x)$. However, shrinking the original relatively to the center of the full membership segment (m), as shown in figure 4.4(ii), guarantees that $\mu_{GREATLY \cdot S}(x) \leq \mu_S(x)$. The following formalises these ideas and redefines the concentration and dilation hedges.

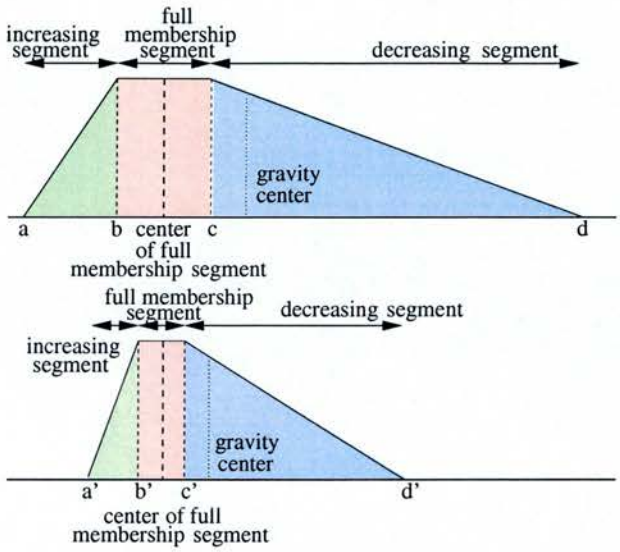


Figure 4.3: Parts of a trapezoidal set and application of the hedge VERY

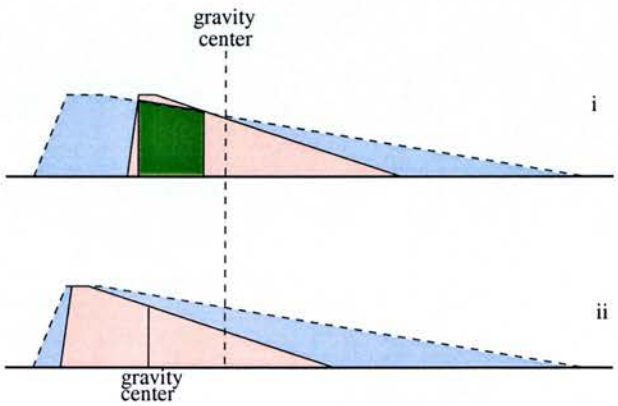


Figure 4.4: Hedge GREATLY: (i) related to center of gravity and (ii) related to center of full membership segment

4.2.1 Concentration

Concentration hedges reduce the size of segments or each part of the membership values of an original set. For a given trapezoidal fuzzy set S with a membership function $\mu_S(x)$ that is characterized by parameters (a,b,c,d), the set modified by a concentration hedge CON should comply with $\forall x \in X, \mu_{CON \cdot S}(x) \leq \mu(x)$. The

parameters of the modified set (a', b', c', d') are therefore defined to be calculated by:

$$\begin{aligned} a' &= b' - ((b-a) * \beta) \quad c' = m + ((c-m) * \beta) \\ b' &= m - ((m-b) * \beta) \quad d' = c' + ((d-c) * \beta) \\ & m = \tfrac{b+c}{2} \end{aligned} \tag{4.1}$$

where β controls the shrinking degree of a concentration hedge. In order to reduce the set effectively β must satisfy $0 < \beta < 1$. In particular, the commonly used hedge terms *MORE*, *VERY* (see figure 4.3) and *EXTREMELY* are defined as follows:

- *MORE* reduces the segments to $\frac{2}{3}$ of the original size ($\beta = \frac{2}{3}$).
- *VERY* reduces the segments by half ($\beta = \frac{1}{2}$).
- *EXTREMELY* reduces the segments to $\frac{1}{8}$ of the original size ($\beta = \frac{1}{8}$).

4.2.2 Dilation

Dilation hedges increase the size of segments or each part of the membership values of a fuzzy set. As opposite to a concentration hedge, a dilation hedge *DIL* should comply with the following intuition: $\forall x \in X, \mu_{DIL \cdot S}(x) \geq \mu_S(x)$. The parameters of the modified set are calculated as concentration hedges, but this time the factors will be greater than one:

- *GREATLY* increases the segments by 2 times the original size ($\beta = 2$).
- *LESS* increases the segments by $\frac{3}{2}$ of the original size ($\beta = \frac{3}{2}$).

It is easy to see that the pair *MORE* and *LESS* and the pair *VERY* and *GREATLY* are complementary; they cancel each other as they express exactly opposite concentration-dilation concepts. A basic optimisation is to remove these pairs if they appear in a model to modify a common fuzzy set. No hedge is found in the literature that matches the opposite of *EXTREMELY* (perhaps *REMOTELY* could be a candidate for this), but including such a dilation hedge is as simple as making $\beta = 8$.

4.3 Newly Introduced Detailisation Hedges

Existing hedges modify a given fuzzy set but cannot create sets which have a focus on certain parts of the original without destroying the original set itself. In practical applications, there are cases where fuzzy models may well be better represented to suit the problem at hand if a loosely predefined fuzzy set could be described with different focuses, say on its lower, middle or upper part. Having recognised this, a new type of three hedges is proposed here, collectively referred to as detailisation hedges. Jointly, they allow decomposing the original set into three fuzzy sets, whilst keeping the order of these decomposed sets the same as the order of the elements belonging to the full membership segment of the original.

Given a trapezoidal set S, characterised by parameters (a,b,c,d), the resulting three sets are denoted by $LOWER\cdot S, MID\cdot S$ and $UPPER\cdot S$, arranged in an increasing order, and they are defined as follows (assuming each is equally characterised by parameters (a',b',c',d)):

$$LOWER\cdot S \begin{cases} a' = a & b' = b \\ c' = b + \frac{c-b}{3} & d' = b + \frac{2*(c-b)}{3} \end{cases} \tag{4.2}$$

$$MID\cdot S \begin{cases} a' = b & b' = b + \frac{c-b}{3} \\ c' = b + \frac{2*(c-b)}{3} & d' = c \end{cases} \tag{4.3}$$

$$UPPER\cdot S \begin{cases} a' = b + \frac{c-b}{3} & b' = b + \frac{2*(c-b)}{3} \\ c' = c & d' = d \end{cases} \tag{4.4}$$

4.4 Traditional Hedges

To ease comparison with the use of conventional hedges, a brief summary of classical dilation/concentration hedges is given below, together with another type of hedge that may be applied to restrict extreme values of linguistic variables.

4.4.1 Dilation/Concentration

As indicated previously, traditional hedges usually use powers of the normal membership functions for dilation and concentration. That is, the traditional form of such hedges is, in general:

$$\mu_{INT\cdot S}(x) = \mu_S^e(x),\ e > 1 \tag{4.5}$$

$$\mu_{DIL\cdot S}(x) = \mu_S^e(x),\ e < 1 \tag{4.6}$$

The particular e may vary. In this chapter the values used for the e of the traditional hedges will be the reciprocal of the β factors of the newly proposed hedges. For example, for the hedge *EXTREMELY* the value of β is $\frac{1}{8}$ so the exponent e for the same *EXTREMELY* hedge used in the traditional way is $e = 8$. Any traditional concentration or dilation hedge can be converted to the newly proposed ones by using β as the reciprocal of the exponent e. Note that β and e are not mathematical counterparts or in any way related. Such conversion from one to the other happens to be a convenience in implementation.

4.4.2 Restriction

Restriction hedges [Cox 94], *ABOVE* and *BELOW*, are applied to variables where fuzzy values are ordered. The set modified by applying the *ABOVE* hedge denotes the set which is "greater than" the original set and that modified by the *BELOW* hedge represents the set which is "less than" the original. The resulting sets are therefore shouldered ones, the left shouldered for *BELOW* and the right shouldered for *ABOVE*. Their membership functions are defined as follows:

$$\mu_{ABOVE \cdot S}(x) = \begin{cases} x < c & 0 \\ c \leq x < d & 1 - \mu_S(x) \\ x \geq d & 1 \end{cases} \tag{4.7}$$

$$\mu_{BELOW \cdot S}(x) = \begin{cases} x < a & 1 \\ a \leq x < b & 1 - \mu_S(x) \\ x \geq b & 0 \end{cases} \tag{4.8}$$

Note that the application of restriction hedges to some fuzzy sets may make no sense. This includes cases where the hedge *ABOVE* is to be applied to a right shouldered set (or any set whose $\mu(x) = 1$ when $x \to \infty$) and similar ones where *BELOW* is used to modify a left shouldered set. Such nonsense hedge-set combinations should be disallowed in modelling.

4.5 The NOT operator

Although *NOT* is not a hedge but a logical operator, in terms of its application effects it may be viewed as a hedge. This is because the application of this operator to a fuzzy set also changes the shape of the membership function of that set (as $\forall x \in X, \mu_{NOT \cdot S}(x) = 1 - \mu_S(x)$). For this reason, it will be treated similarly as any other hedge hereafter.

4.6 A Final Note on Hedges

In applying the above defined hedges, for any original shouldered membership function, it is assumed that the original fuzzy set is trapezoidal with its extreme parameters (a and b for the left shouldered and c and d for the right shouldered) being equal to a certain domain maximum/minimum value. These limit values can be determined using domain knowledge (e.g. by identifying the maximum and minimum values in the training data set with, perhaps, a safe offset of, say, $\pm 10\%$). If these

maximum/minimum values get updated (for example, when new training data becomes available), the same hedge applied to a shouldered set before or after the update may yield different modified sets. In the event that the new maximum/minimum value exceeds the corresponding safe offset, modifications made to shouldered sets must then be recomputed using the new limit values.

To illustrate and compare the effects of applying the hedges conventionally used and presently proposed, figures 4.5 and 4.6 are herein included. Figure 4.5 shows the results before and after a hedge is used to modify an irregular piecewise linear fuzzy set using the traditional hedges. Figure 4.6 gives further illustration of such results.

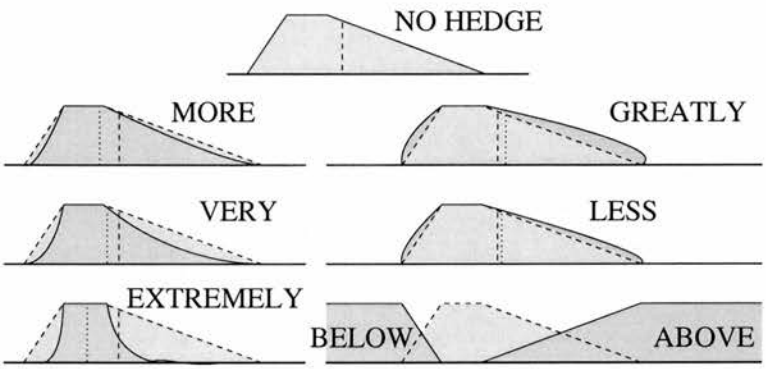


Figure 4.5: Traditional hedges applied to an irregular trapezoid and how they change the center of gravity

4.7 Summary

A description of the hedges used in this thesis have been given in this chapter. The definitions of classical hedges were introduced, and the newly proposed hedges explained. A novel method of carrying out the modifications that the hedges perform on the fuzzy sets was also proposed.

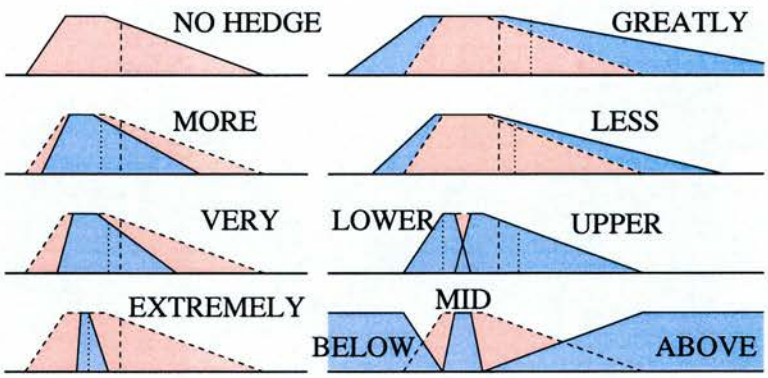


Figure 4.6: New proposed hedges applied to an irregular trapezoid and how they change the center of gravity

Chapter 5

The Framework for Translation

"We are all agreed that your theory is crazy. The question which divides us is whether it is crazy enough to have a chance of being correct."

Niels Bohr

5.1 Introduction

The main aim of this work is to find an efficient and effective way to translate rules that use approximative sets into rules that use descriptive sets and hedges. The translated rules will be functionally equivalent to the original, or a close equivalent within the limitations of the GA search, with the advantage of having or regaining human-comprehensible interpretation. It does not matter as to which technique is used to generate the original approximative rules. All that is required is a set of approximative rules and the definition of the descriptive fuzzy sets and linguistic hedges. In the experimental studies to be presented later, a hybrid method is used to produce the initial approximative rules. Trapezoids are herein adopted as final descriptive sets for computational efficiency purposes.

To perform the mapping a concatenation of two methods is proposed here. One is based on a heuristic search. (As the space of potential descriptive rules can be very

large, techniques of branch and bound are applied and so the power of this heuristic method may be rather restrictive.) The other uses a GA to work on the full search space. As evolutionary search usually works better when a good start point has been identified [Fogarty 96, Kallel & Schoenauer 97], the first method will be employed as the generator of the initial population for the GA, which will then make a finer-grain search. The heuristic translation may not yield spectacular results but it is far better than a random start as will be shown later. Chapter 6 will cover the heuristic translation in detail.

The evolutionary computation-based approach proposed here depends on the concept of *functional equivalence*. It works by searching for a set of descriptive rules that collectively behave like the original approximative rule from which they are translated. That is, for data that is covered by an approximative rule, the translated descriptive rule(s) will fire with at least the same firing strength as the original. Furthermore, for data that would not cause the original approximative rule to fire, the resultant descriptive rules will either not fire or fire if their consequents comply with the desired output. As indicated before, the search mechanism is herein implemented by a GA. The following sections cover the more general aspects and implementation independent issues of the translation. Chapter 7 is devoted to the specific mechanisms of the genetic algorithm created for the task of translation.

5.2 Training Sets

Each approximative rule may be translated independently. Multiple descriptive rules are considered per approximative rule as, in general, an approximative rule R may not be covered by just one resulting descriptive rule. To implement the translation in this manner, a training subset for each approximative rule R needs to be generated from the original training set (from which the approximative rules were obtained). Such rule

training sub-sets are derived via a data selection and enrichment process as introduced below.

Suppose that there are K_R emerging descriptive rules that, collectively, form the *functional equivalent* to a given approximative rule R. Given a set X of the original training examples, for each $x_i \in X$ the firing strength $AFS_R(x_i)$ of the approximative rule under translation is calculated. If $AFS_R(x_i) > 0$ it would be desirable that, if the consequent of this rule is the same as the desired consequent, any of the resulting translated descriptive rules $R_j, j = 1, .., K_R$ will fire for such an example with a strength $DFS_{R_j}(x_i)$ equal to or greater than $AFS_R(x_i)$. This kind of example will be hereafter referred to as an example of *type one*. If, however, $AFS_R(x_i) > 0$ and the consequent does not match the desired, then the firing strength of each resulting descriptive rule $DFS_{R_j}(x_i)$ should be less than, or at worst equal to, $AFS_R(x_i)$. This kind of example will be referred to as *type two*. Furthermore, if $AFS_R(x_i) = 0$ then, if the example x_i is of the same desired consequent as that of the original rule, it is not selected to form the training sub-set (as this example provides no influence in executing this learned rule and is expected to be covered by other approximative rules). If, however, the consequent is different, the firing strength of the resulting descriptive rules should be zero. This last kind of example will be referred to as *type three*.

Clearly, for any $x_i \in X$ and a given original approximative rule R, x_i is selected to form the training subset for translating R if and only if it is an example of one of the three types. Each data point is therefore enriched by the inclusion of its type and its $AFS_R(x_i)$.

5.3 Objective Functions

The above data selection process allows the GA to enforce the following objectives in performing search for suitable descriptive rules, where $T_t, t \in 1, 2, 3$, denotes the subset of training data of type t:

$$\forall x_i \in T_1 \quad DFS_{R_j} \geq AFS_R(x_i) \tag{5.1}$$

$$\forall x_i \in T_2 \quad DFS_{R_j} \leq AFS_R(x_i) \tag{5.2}$$

$$\forall x_i \in T_3 \quad DFS_{R_j} = 0 \tag{5.3}$$

As the classification inference is to be performed by choosing the output value of the rule that has the highest firing strength, enforcing the above conditions yields a descriptive model that is at least as accurate as the original approximative model. This is because the inequality restrictions allow an increase in the firing strength of the descriptive rules to be learned over correct training data (type one) and a reduction in the firing strength over incorrect training examples (type two). However, in general, not all training examples will satisfy these restrictions and it is the job for the GA to reduce the discrepancies between the descriptive and approximative firing strengths as much as possible.

5.4 Functional Equivalence Objectives

For efficiency, the GA should be guided to search for a set of emerging descriptive rules of a minimum cardinality. This means that an objective is needed to minimize the number of descriptive rules used to act as the given approximative rule. Also, any difference between the *descriptive firing strength* (DFS) (that is, the maximum firing strength within the resulting descriptive rule set for the data points under consideration) of the emerging rules and the *approximative firing strength* (AFS) of the original approximative rule for each data type should be restricted to minimum. Hence, another objective is introduced to minimize the variance of individual rule error. This way, the error that may be produced by the translated rules will be as much evenly distributed among all rules as possible, thereby avoiding individual rules with a particularly high error.

In summary, in searching for a set of descriptive rules that would jointly function as a given original approximative rule R, the GA search will be guided by objectives as listed in table 5.1, where K_R is the current number of emerging descriptive rules.

Table 5.1: Functional Equivalence (Minimising) Objectives

Expression	*Description*
$\sum_{x_i \in T_1, j=1,\ldots,K_R} max(0, AFS_R(x_i) - DFS_{R_j}(x_i))$	Error of type 1 training data
$\sum_{x_i \in T_2, j=1,\ldots,K_R} max(0, DFS_{R_j}(x_i) - AFS_R(x_i))$	Error of type 2 training data
$\sum_{x_i \in T_3, j=1,\ldots,K_R} DFS_{R_j}(x_i)$	Error of type 3 training data
$\frac{1}{K_r} \sum_{j=1,\ldots,K_R} (\delta_R - E_{R_j})^2$	Overall error variance
K_R	Number of rules

In this table E_{R_j} denotes the individual error of a descriptive rule, and δ_R represents the mean of the individual errors of all the emerging descriptive rules with regard to the original approximative rule R. They are defined as follows:

$$\begin{aligned} E_{R_j} = \; & \textstyle\sum_{x_i \in T_1} max(0, AFS_R(x_i) - DFS_{R_j}(x_i)) + \\ & \textstyle\sum_{x_i \in T_2} max(0, DFS_{R_j}(x_i) - AFS_R(x_i)) + \\ & \textstyle\sum_{x_i \in T_3} DFS_{R_j}(x_i) \end{aligned} \tag{5.4}$$

$$\delta_R = \sum_{j=1,\ldots,K_R} \frac{E_{R_j}}{K_R} \tag{5.5}$$

There exist in the GA literature several approaches to deal with such problems that have multiple objectives, including the aggregation approach [Goldberg 89],

Non-Pareto approach [Schaffer & Grefenstette 85] and Pareto-Based approach [Goldberg 89]. The present work adopts the first of these, as it offers a conceptually simpler method by converting multiple objectives into a compounded single objective. In particular the aggregation function used is the Sum of Weighted Global Ratios (SWGR) [Bentley & Wakefield 96, Bentley 99]. The aggregation method first independently normalizes each objective with respect to the best and worst value ever found for it and, then, weights and adds together each objective to form the single overall fitness value. The structure of the final fitness function will be fully explained in chapter 7 (section 7.6).

5.5 Classification Objectives

Pure *functional equivalence* guidance (i.e., the exclusive use of only the functional equivalence objectives of table 5.1) may miss some otherwise possible improvements of the overall performance of the learned ruleset. This is because of its trying to fit the approximative model rather than to fit the training data. Empirically, for classification problems, it is generally better to use the influence of *functional equivalence* objectives along with additional classification objectives as provided in table 5.2 and to decrease *functional equivalence* influence as the GA goes on. That is, the search will initially have a strong focus on the improvement of the heuristic translation to get close to the approximative model and later it will concentrate on the satisfaction of the classification-specific objectives. The reason that search is not guided by the classification objectives alone right from the start is to speed up the finding of optimal descriptive ruleset, by first approximating the emerging descriptive rules to a potentially good accuracy level (offered by the good approximative model) and then optimizing them locally.

Finally, it is worth noting that no guarantees may be given to obtain the closest equivalent translation when a GA run terminates. In general, such a guarantee

Table 5.2: Additional Classification Objectives

Maximize Number of Correctly Classified
Minimize Number of Incorrectly Classified
Minimize Number of Not Covered

cannot be obtained without performing exhaustive search. However, given limited computational resources, the translations are empirically very close to the original approximative model in function (as shown later).

5.6 Strategies for Rule Translation

The individual translation strategy described in section 5.2 has the drawback that, when the individual translations are put together to form the final translated descriptive rule set, the independently translated rules may interfere with each other. Although a close fit of the descriptive rules to the approximative ones may help resolve this problem, this cannot be guaranteed. It is therefore interesting to consider possible alternative translation strategies.

An example of individual translation interference can be seen in figure 5.1. Three approximative rules (ellipsoidal shaped), two belonging to class $\times$ and one to class $+$, are translated using individual strategies. Although some of the descriptive rules (boxes) contain elements belonging to a different class the error may be acceptable due to pressure of obtaining the least number of rules possible. The blue box that translate the class $+$ is a perfect translation. When all three rules are put together they interfere among themselves.

Instead of translating individual approximative rules one by one, the first possible alternative approach, valid for problems with a limited range of output values (such

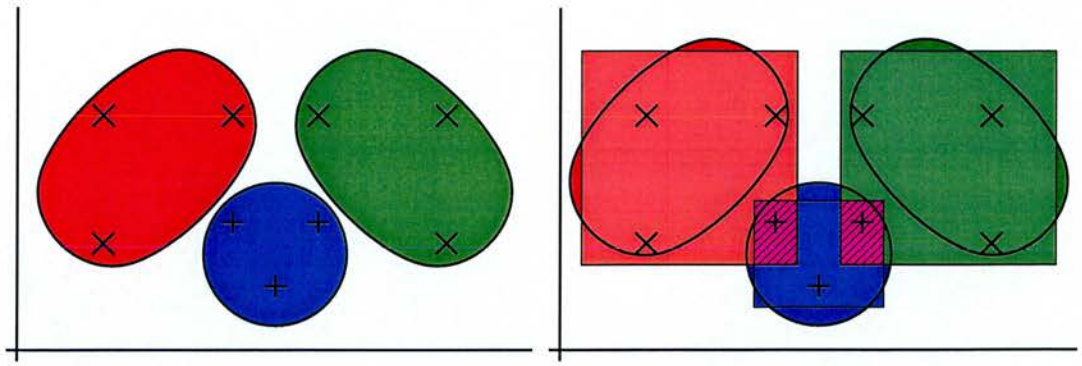


Figure 5.1: Interference of individual translation

as classification problems as mainly concerned herein), is to translate a group of all the approximative rules regarding a single output value at a time. In this so-called group strategy an AFS value is calculated as the firing strength of the entire subset of rules concerning the same output value, which is defined by the strongest firing strength of all the original approximative rules that characterize the same class. Thus the translation can be done class by class, instead of rule by rule.

An example of how this second strategy can help the translation process is shown in figure 5.2. Here, a third rule can prevent the $\times$ rules to misclassify some $+$. To obtain a similar effect in individual strategy four rules would be needed.

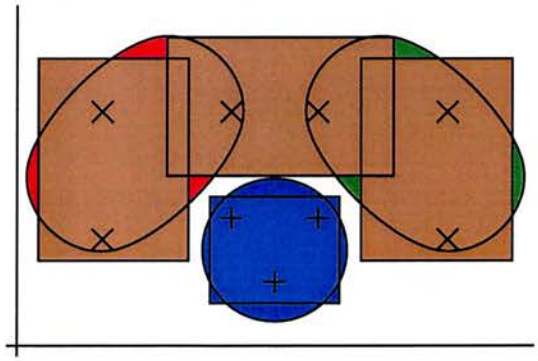


Figure 5.2: Possible optimization via group translation

For completeness, another version of the GA search strategy is also included here, and termed the global strategy, where all rules for all classes are represented together in each chromosome. That is, a chromosome is itself a whole translation of the given approximative model, following the Pittsburgh style GAs [Smith 80].

In summary, the strategies available for implementing the translation are:

- **Individual translation:** Each rule is translated independently. It requires a new GA to be run for each rule to be translated. Only the antecedents of the rules have to be encoded in the chromosome.

- **Group translation:** Several rules are translated at the same time. The rules are grouped by involving the same output. It requires a GA to be run for each identified group of rules to perform the complete translation of the model. Again, only the antecedents of the rules have to be encoded in the chromosome, as all rules within one group share the same output.

- **Global translation:** All the approximative rules are translated at the same time. The chromosome must, therefore, encode the output of the rules. It requires the GA to be run only once in order to obtain the complete translation.

5.7 Summary

This chapter has proposed the main framework for translating approximative models to descriptive ones. The methods for upgrading the training sets with new information needed for the translation were described. The objective functions used by the GA were shown and different strategies that can be followed to implement the translation were given.

Chapter 6

Heuristic Methods

"Common Sense is the least common of the senses."

Spanish Proverb

6.1 Introduction

Heuristic as an adjective, pertains to the process of gaining knowledge or some desired result by intelligent guesswork and common sense rather than by following some pre-established formula in contrast with algorithmic. It can be used to describe an approach to learning by trying without necessarily having an organized hypothesis. That is, "trial-by-error" learning. It may also refer to the use of the general knowledge gained by experience, sometimes expressed as "using a rule-of-thumb". As a noun, a heuristic is a specific rule-of-thumb or argument derived from experience.

This chapter explains a general and two extended methods that use rules-of-thumb. These methods are developed to help translate approximative rules into descriptive ones. Each approximative rule defines an area, a hyperbox in the input space that behaves in a similar way or leads to the same output. The general idea that guides the development of these heuristic methods is to find which combination of descriptive hyperboxes covers the area defined by the approximative hyperbox. It is obvious to

see that the more similar a descriptive hyperbox (or a combination of them) is to the approximative one the better the translation will be. As the number of potential different descriptive rules can be huge (see equation (2.60)) and as hedges introduce many more different possible descriptive hyperboxes (see equation (2.65)), many of them overlapping, a method to preselect combinations of rules will be of great use. Here are explained several of these methods.

6.2 The Basic Heuristic Approach

This approach is based on hyperbox similarity with given approximative rules. Descriptive rules, i.e. hyperboxes defined on descriptive sets, are created if they are similar (in fact if they intersect) with an approximative rule (or the hyperbox defined by the antecedent approximative sets). This produces a preliminary translation, which is the one often used for explanation purposes in the existing literature [Sugeno & Yasukawa 93]. The basic component of this proposed method uses no hedges and serves to introduce the underlying ideas of the heuristic translation. The extensions will be explained in the following sections.

6.2.1 Similarity graph

The basic heuristic method works by building a layered graph to represent degrees of similarity between approximative and descriptive sets, using an intersection-based similarity measure. Each layer of the graph consists of a certain number of nodes; each of which represents the degree of similarity between one of the approximative sets of an antecedent variable and one of the descriptive sets of the same variable. Thus, each path of the resulting graph may be interpreted as a possible descriptive rule which coarsely approximates a given approximative one. The amount of similarity between two sets S_1 and S_2 is hereafter called the *Similarity Value* (I) of the two. In this work

the similarity used to build the similarity graph is defined by:

$$I_{S_1S_2} = \frac{A(S_1 \cap S_2)}{A(S_1)} * \frac{A(S_1 \cap S_2)}{A(S_2)} \tag{6.1}$$

with $A(Set)$ denoting the area of the set Set. The $A(Set)$ is calculated, in practice, as the integral of the membership function over its definition domain.

Note that similarity measures represent the degree to which two fuzzy sets descriptions coincide. The similarity value may vary from zero for no intersection to one for equality.

When similarity is smaller than a given minimum value, termed as I-threshold, the corresponding node is removed from the graph. Incidentally, although the above particular definition is utilized in this thesis, empirical results have shown that other similarity metrics proposed in the literature [Yeung & Tsang 97, Zwick *et al.* 87, Zadeh 87] may be adopted to take its place without major disruption in the mapping results. However, this definition has proven to be computationally simple and performance-wise robust.

Supported by such a similarity metric, given an approximative rule Q:

$$\text{IF } x_1 \text{ is } S_1 \text{ AND } x_2 \text{ is } S_2 \text{ AND } \ldots \text{ AND } x_p \text{ is } S_p \text{ THEN } Class$$

and a collection of descriptive sets $\{L_{ij} \mid j = 1, 2, \ldots, k_i\}$ per variable x_i the preliminary method to build the graph is summarized in figure 6.1.

6.2.2 An example of similarity graph generation

To illustrate this basic approach consider the following example. Assume that the input space is two-dimensional. For each of the two input variables, x_1 and x_2, three descriptive fuzzy sets are defined such that x_1 may take a value on either L_{11} =Low, L_{12} =Medium or L_{13} =High, and x_2 on either L_{21} =Small, L_{22} =Medium or L_{23} =Large. Suppose that the approximative rule to be translated is:

$$\text{IF } x_1 \text{ IS } S_1 \text{ AND } x_2 \text{ IS } S_2 \text{ THEN } A$$

```
for i = 1 to p
  for j = 1 to |L_i|
    if I_{S_i L_ij} > I-threshold
       add(generate_node(i,j), graph)
    endif
  endfor
endfor
for i = 2 to p
  for j = 1 to |L_i|
    for k = 1 to |L_{i-1}|
      connect(node(i,j),node(i-1,k))
    endfor
  endfor
endfor
reset_find_path(graph,1,p)
while (more_paths(graph,1,p))
  add(generate_rule(get_next_path(graph,1,p)),rulebase)
endwhile
```

Figure 6.1: Graph generation algorithm

where S_1 and S_2 are two approximative fuzzy sets defined on the domains of x_1 and x_2 respectively and A is a possible class value. The descriptive sets, the grid generated by these sets and the hyperbox covered by the approximate rule are shown in figure 6.2.

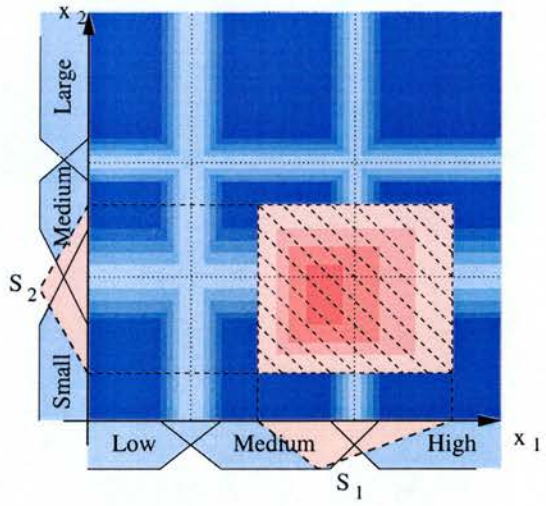


Figure 6.2: Approximative rule and descriptive sets

The first layer of nodes is created by taking on the approximative set of the first antecedent of the original rule, in this case S_1, and then by constructing a node for each descriptive set L_{1i}, $i = 1,2,3$ (that is, Low, Medium and High) of x_1 if the similarity measure between L_{1i} and S_1 is larger than the I-threshold (zero by default). Suppose that the measure between S_1 and L_{12} and that between S_1 and L_{13} are $I_{S_1L_{12}} = 0.62$ and $I_{S_1L_{13}} = 0.43$, respectively. Also, suppose that S_1 does not intersect L_{11} (that is, the similarity is zero). Thus, two nodes are created, as illustrated under S_1 in figure 6.3.

This process is repeated for each variable that appears in the antecedent of the original approximative rule, resulting in different layers of nodes with each layer corresponding to one variable. Then, all the nodes in one layer are connected to the nodes in the next with the arrow of each link pointing from a previous node to a newly created one, as also shown in figure 6.4.

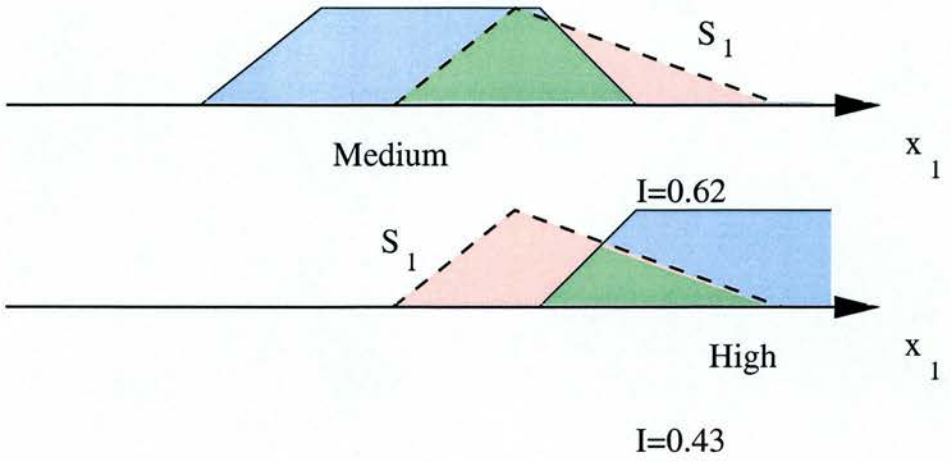


Figure 6.3: Similarities of descriptive and approximative sets

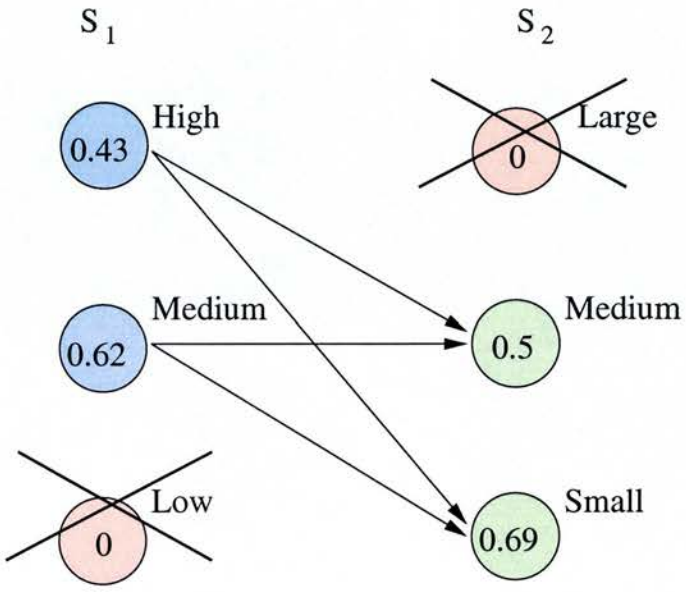


Figure 6.4: Graph generation

Once the graph is generated paths from any node in the first layer to a node in the last are constructed. Each path becomes an emerging rule, with the antecedent variables taking the labels of the nodes of that path. Thus, the resultant set of descriptive rules which collectively form a preliminary translation of the given approximative rule are:

R_1: IF x_1 IS High AND x_2 IS Medium THEN A

R_2: IF x_1 IS High AND x_2 IS Small THEN A

R_3: IF x_1 IS Medium AND x_2 IS Medium THEN A

R_4: IF x_1 IS Medium AND x_2 IS Small THEN A

6.3 Extending Basic Heuristic Method

The basic heuristic method given above does not ensure a good coverage of an approximative rule unless the I-threshold used is very low. However, a low threshold potentially implies many nodes and hence many descriptive rules. This implies that the method can be quite sensitive to such parameter settings. Nevertheless, this method is proposed to act as the starting point for the evolutionary search and its accuracy is not of utmost importance. Also, it can be itself improved.

An obvious improvement is to include hedges. In so doing, the number of nodes will, however, increase drastically. This is because the label of a node may now be any combination of a descriptive set and a number of hedges used to modify the set. Even if nodes with a similarity value below the I-threshold are eliminated, and if the up-ceiling of the number of hedges applicable to a set is limited to two, this may still result in a significant increase of the number of nodes in the graph.

To reduce such increases and hence the number of emerging descriptive rules, various heuristics may be applied to eliminate unwanted nodes. In particular, if some nodes within a layer are similar to each other only one of them would then be needed. That is, for each layer, a similarity measure between any pair of nodes is calculated. Those nodes that are very similar to the others can then be eliminated providing that at least one of them is retained, as the space they cover, over the approximative set, will be approximately covered by the surviving similar fellow nodes.

Also, external control of the desirable distinctions amongst possible values per input variable, that is the number of nodes permitted per layer, can be used to select

those which are most dissimilar between one another. Both methods are implemented in this work; they ensure that the nodes left are different among themselves. Of course, these methods are supported by satisfying the requirement that whatever nodes to be chosen they must attain a high similarity value.

In summary, for this elimination process two different heuristic methods were used. The first ensures that all nodes will not be similar among themselves above a certain threshold (termed S-threshold). The second imposes a limit over the number of nodes per layer. These are detailed below.

6.3.1 Extended heuristic method 1

The first method (Extended Heuristic 1 or EHeu 1 for short) works by generating a similarity matrix. For each layer a similarity measure between any pair of nodes is calculated yielding the matrix. It is easy to see that this matrix will be symmetrical and will have a diagonal of ones (as each set is completely similar to itself).

An example, assuming the use of only one hedge per modification, is given in table 6.1. In this example the descriptive sets Medium and High and the original approximative set A to be translated are defined as follows: Medium=(2,4,8,10), High=(8,10,14,16) and A=(3,9,14,15). The similarity values between each descriptive set (with or without hedge modification) and the original set A are also shown in this figure. Only some significant combinations of hedge-set have been selected for clarity purposes. The similarity measure used between a pair of fuzzy sets S_1 and S_2 is:

$$S_{S_1S_2} = I_{S_1S_2} = \frac{A(S_1 \cap S_2)}{A(S_1)} * \frac{A(S_1 \cap S_2)}{A(S_2)} \tag{6.2}$$

Note that this similarity measure is equal to the one used in equation (6.1). Any similarity measure would do, of course.

Once the similarities are calculated those cells whose similarity value is higher than

a given S-threshold are marked. In figure 6.3.1 such cells, for an S-threshold of 0.25, are marked as crossed ones.

Table 6.1: Similarity Matrix

	Great M	Ver M	High	Great H	Ver H	Ext H
Greatly Medium	1	0.25	0.2	0.38	0.12	0.04
Very Medium	0.25	1	0	0.14	0	0
High	0.2	0	1	0.6	0.5	0.12
Greatly High	0.38	0.14	0.6	1	0.3	0.07
Very High	0.12	0	0.5	0.3	1	0.25
Extremely High	0.04	0	0.12	0.07	0.25	1

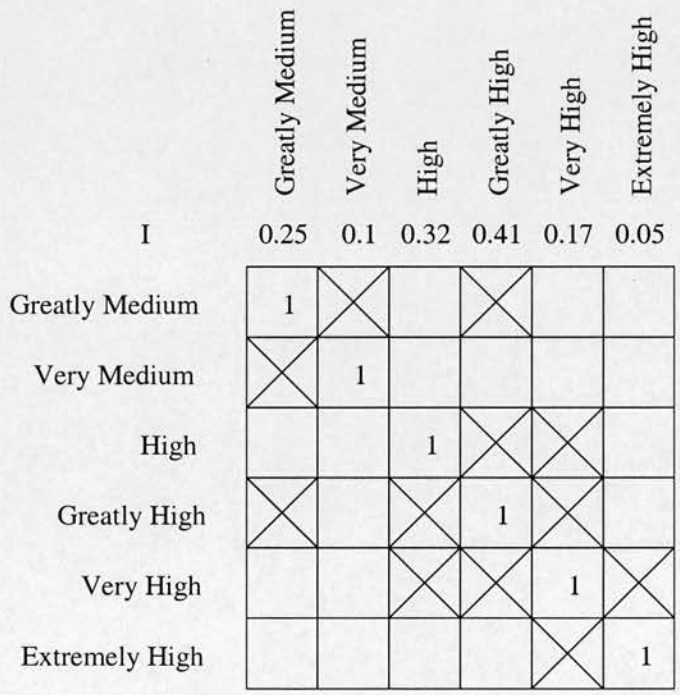


Figure 6.5: Example of similarity matrix (crosses represent an unacceptable similarity)

6.3.1.1 Preserving best I-similarity

Having obtained a marked similarity matrix, the next step is to eliminate certain columns and rows of the matrix to obtain another dimensionally reduced matrix with no crosses on it, that is, with elements whose mutual similarities are not greater than the given S-threshold. The elimination is implemented by:

a) Identifying the node that is associated with the highest or best I value and that has a similarity value above the S-threshold with other nodes (that is, has crosses on its columns/rows);

b) Removing other nodes similar to the identified node (above the S-threshold, of course), as they have a lesser I value; and

c) Iterating this process for the node of the next highest I value which possesses a similarity value still above the S-threshold in comparison to other remaining nodes.

This strategy has an intuitive appeal in that the nodes with the highest I values are more likely to capture what is embedded in their respective parts of the original approximative rule than their similar fellow nodes.

An example of applying this strategy is also presented in figure 6.6. The node Greatly High is first identified as the one having the best I-similarity value. Those nodes that have a cross with it are then deleted, thereby eliminating Greatly Medium, High and Very High. After this there are no more crosses in the similarity matrix so the process terminates. Note that the numbers on the crossing outs tell the step in which they are eliminated (in this case all of them in the first step). This results in a dimensionally reduced matrix as given in table 6.2. Thus, three remaining hedge-modified sets: Very Medium, Greatly High and Extremely High will be used to describe the approximative set A of the antecedent variable x in the original rule under translation.

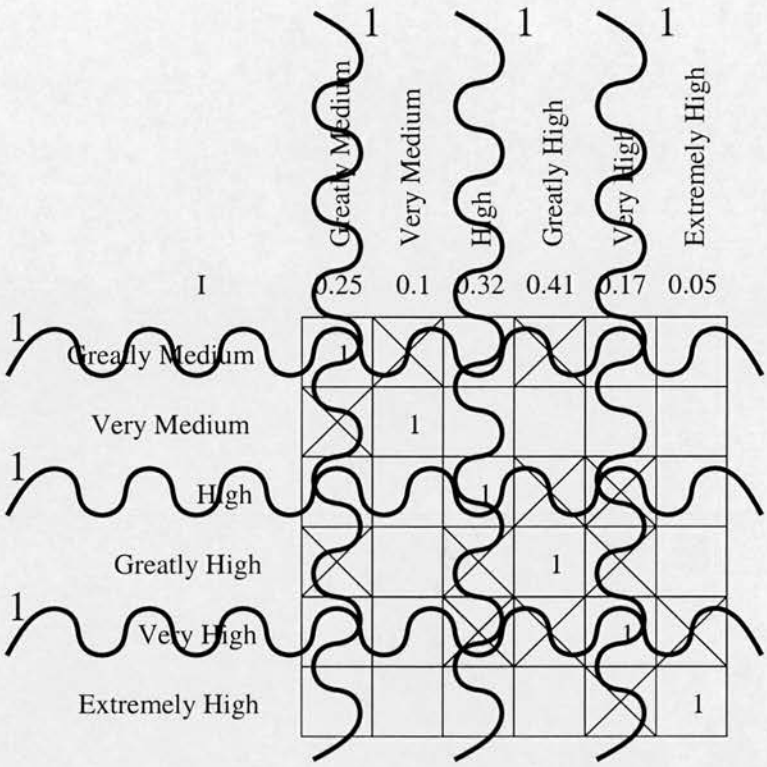


Figure 6.6: Simplification of the similarity table with the "preserving best I" strategy

Table 6.2: Dimensionally Reduced Similarity Matrix with the "preserving best I" strategy

	Ver M	Great H	Ext H
Very Medium	1	0.14	0
Greatly High	0.14	1	0.07
Extremely High	0	0.07	1

6.3.1.2 Eliminating sets included in others

The example given above can be used to explain another modification performed on the layers of the graph. Some of the fuzzy sets of a graph layer may be contained (fully included) in a fellow set within the same layer. For example, Extremely High

is contained inside High (Extremely is a concentration hedge and is always inside the set it modifies, look figure 4.6) although they are dissimilar. This means that whatever is covered by Extremely High is also covered by High. It seems natural to remove Extremely High as rules using such a set will be inside the ones using High, and then the points would be covered by two rules, one being a more general version of the other (in this case the one using High would be more general than the one using Extremely High). The problem with this is that perhaps High does not fit the approximative set as well as Extremely High.

In a sense, the I value can be interpreted like a density, the amount of the approximative set that is dissolved in the descriptive set. If, for example, Extremely High has an I value of, say, 0.8 and High has an I value of only 0.2, then the more general rule overcovers the approximative set, that is, the approximative set is more "diluted" within the rule, while the more specialized fits better, the approximative set is more "concentrated". In this case it would not be interesting to eliminate the included descriptive set as it is more close to the approximative.

However, if it is the specialized set the one with a smaller I value then the approximative set is more present in other zones of the general set than in the specialized one. It is, therefore, interesting to remove the specialized set as the general one covers all the zones that the specialized covers and does in a better way. Considering these cases, the following is adopted for all the nodes N_{ij} of a layer G_i.

$$\forall j,k \in \{1,\ldots,|G_i|\}, j \neq k \mid N_{ij} \subseteq N_{ik} \quad \text{if } I_{ij} < I_{ik} \text{ remove } N_{ij} \tag{6.3}$$

6.3.1.3 Removing most crosses

Another strategy potentially useful for supporting the EHeu1 method is to remove the nodes with most crosses in their corresponding row/column, thereby eliminating those that are similar to many others. The rationale behind this strategy is that it gets a more

even coverage of the whole approximative set. However, the strategy tends to remove fewer nodes and hence may cause more emerging descriptive rules to survive.

The outcome of applying this strategy can be seen in figure 6.7 for the present example. In this case the nodes with the largest number (3) of crosses are Greatly High and Very High. Given this tie situation the one with lower I is deemed to lose and, therefore, Very High is eliminated. Now, Greatly High involves two crosses and is the one with the most crosses attached so it is eliminated also. After this, only a cross is left between Very Medium and Greatly Medium. As Very Medium has a lower I value, it is deleted. Table 6.3 shows the reduced matrix. Thus, three remaining hedge-modified sets; Greatly Medium, High and Extremely High will be used to describe the approximative set A of the antecedent variable x in the original rule under translation.

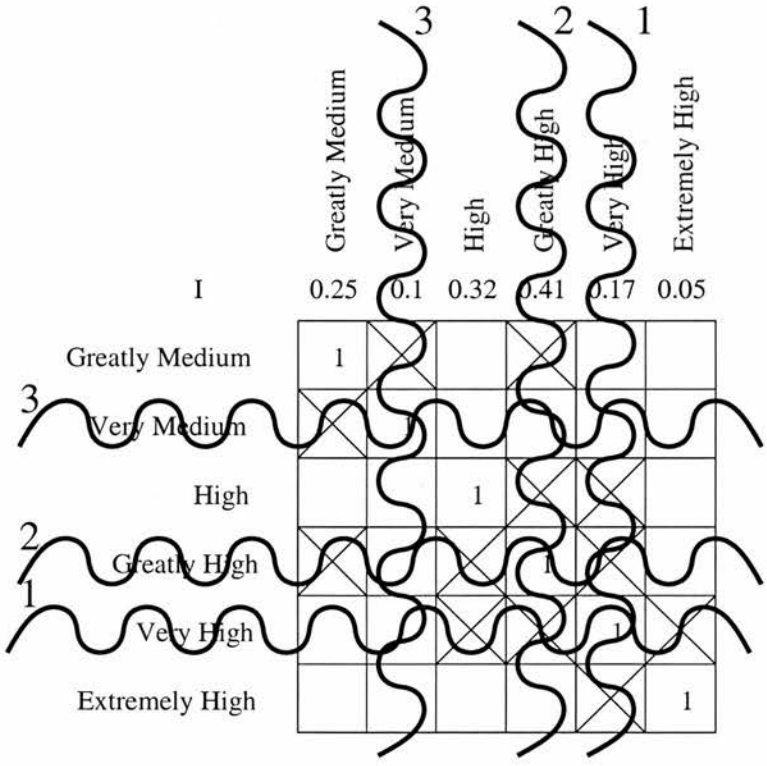


Figure 6.7: Similarity matrix simplification with the "removing most crosses" strategy

The above two strategies may be integrated using a linear combination of the best I-similarity value and the number of crosses in order to identify those nodes that can be removed to produce efficient translations.

Table 6.3: Dimensionally Reduced Similarity Matrix with the "removing most crosses" strategy

	Great M	High	Ext H
Greatly Medium	1	0.2	0.04
High	0.2	1	0.12
Extremely High	0.04	0.12	1

6.3.2 Extended heuristic method 2

The second extended heuristic method (EHeu 2 for short) works by imposing a constraint over the number of nodes per layer via a procedure that:

a) Sorts the cells of the matrix from the largest to the smallest similarity values, figure 6.3.2 showing the sorting result for the current example;

b) Visits each cell in the sorting order and, of the two nodes that define the cell, deleting the one with a smaller I; and

c) Iterates the process until only an allowed number of nodes remain.

If, for instance, the limiting number of nodes to keep is 3 the above procedure works in the illustrative example as follows. The cell with the highest similarity is defined by Greatly High and High. As High has a smaller I value, it is deleted. The cell of the second highest similarity value would have been defined by the pair High and Very High but High has already been eliminated and so this cell is ignored. Without High the next cell determined in the sorting order is the one marked by 3 and defined by

	Greatly Medium	Very Medium	High	Greatly High	Very High	Extremely High
I	0.25	0.1	0.32	0.41	0.17	0.05
Greatly Medium		5	7	3	10	12
Very Medium				8		
High				1	2	9
Greatly High					4	11
Very High						6

Figure 6.8: Sorted cells defined by pairs of nodes (the integers represent the ranks of the sets after sorting)

Greatly High and Greatly Medium, Greatly Medium has a smaller I value than Greatly High and therefore is removed. Finally, the cell with ranks 4 indicates that Very High can be deleted. As there are only 3 nodes remaining the algorithm stops. The process terminates with the following three hedge modified nodes remaining: Very Medium, Greatly High and Extremely High.

6.3.3 Improving extended heuristic method 2

The above method can be improved further. In sorting exclusively by similarity and using I just to decide which node of a pair to eliminate it relies largely on the similarity measures. However, if the two nodes defining a cell have both a high I value it would be interesting to shift its place in the sorting list of all the cells to a later position in

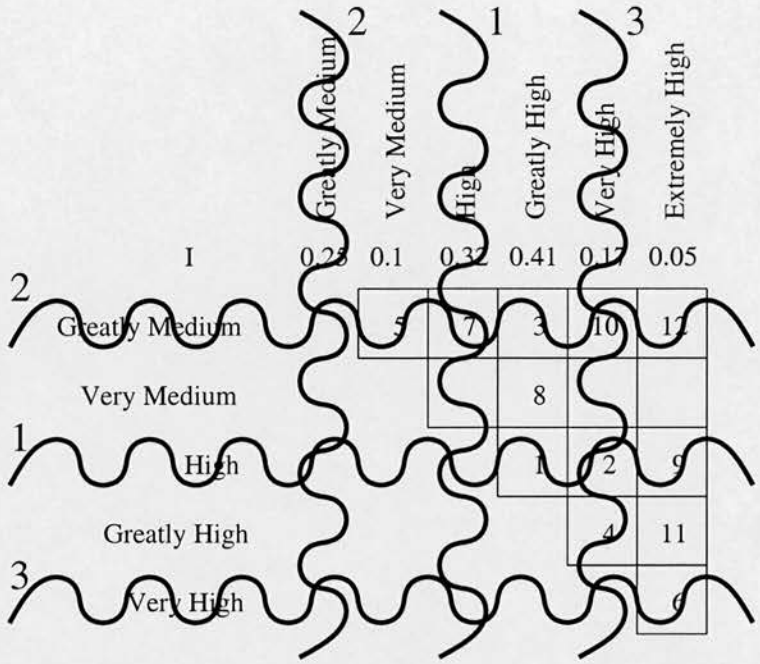


Figure 6.9: Nodes eliminated using Extended Heuristic 2

order that, when the elimination process stops, surviving nodes will still have a high I-similarity value, whilst being dissimilar to other remaining nodes. For this purpose, the original similarity matrix (e.g. the one given in table 6.1) can be modified by subtracting from each cell the average of the I values of the two nodes that define that cell. That is, given two nodes N_i and N_j, $i \neq j$, $i, j \in \{1, 2, \ldots, k\}$ with k being the number of nodes in a particular layer say, layer m, the modified similarity value between them is calculated by:

$$S(N_i, N_j) - \frac{(I_{N_i A_m} + I_{N_j A_m})}{2}$$

For the example, the modified similarity matrix, as the counterpart of that of table 6.1, is given in table 6.4. From this, nodes are re-sorted, resulting in the ordered cells in figure 6.10. Note that, as the I values decrease very fast as the dissimilarity between

two nodes increases, the modified similarity matrix, and hence the resorted cells, will not be drastically different from their original.

Table 6.4: Modified Similarity Matrix

	Great M	Ver M	High	Great H	Ver H	Ext H
Greatly Medium	0.75	0.08	-0.09	0.04	-0.09	-0.11
Very Medium	0.08	0.90	-0.21	-0.12	-0.14	-0.07
High	-0.09	-0.21	0.68	0.23	0.25	-0.06
Greatly High	0.04	-0.12	0.23	0.59	0.01	-0.15
Very High	-0.09	-0.14	0.25	0.01	0.82	0.14
Extremely High	-0.11	-0.07	-0.06	-0.15	0.14	0.96

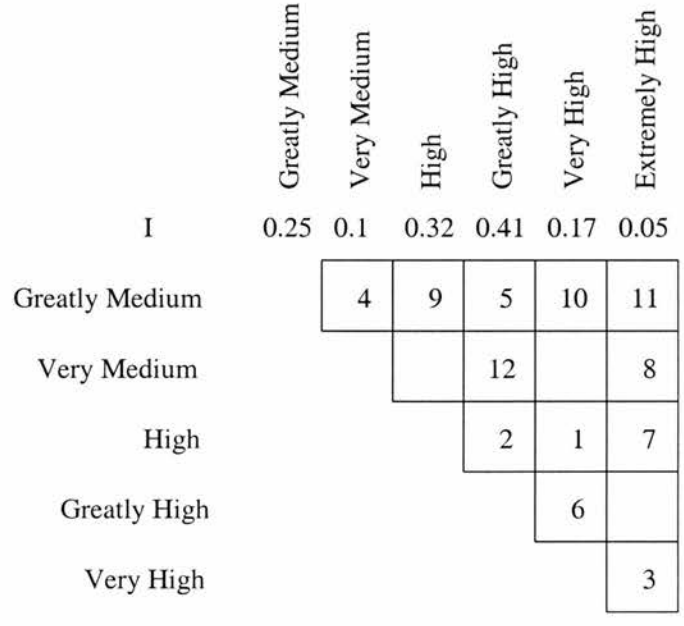


Figure 6.10: Sorted cells using modified similarity matrix

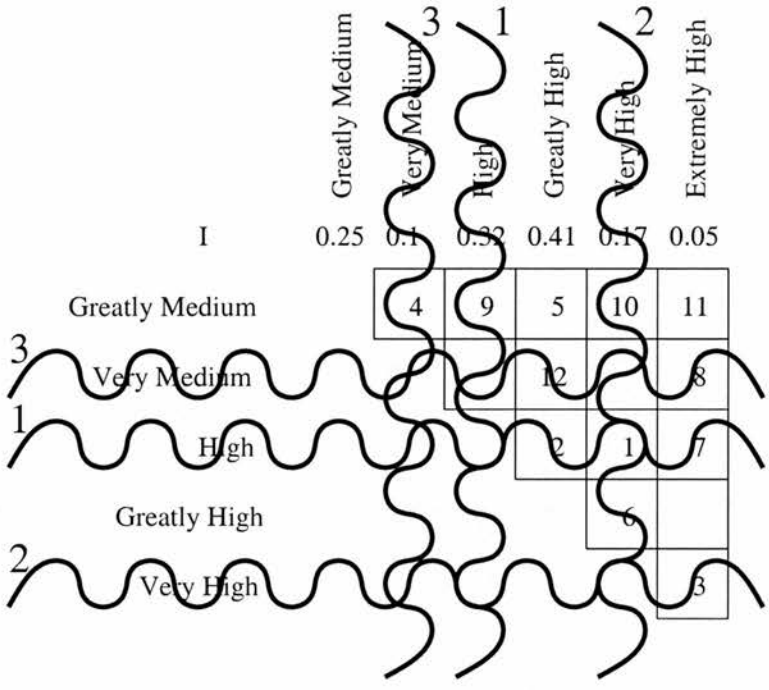


Figure 6.11: Nodes eliminated on modified similarity matrix using Extended Heuristic 2

Using the re-sorted cells for the present example the order of node removal becomes: Very High, then High and finally Very Medium. This preserves the following nodes: Greatly Medium, Greatly High and Extremely High.

6.4 Summary

This chapter is devoted to the explanation of the working mechanisms of the heuristic methods used to obtain an initial translation, for further search-based methods to modify. A basic translation was illustrated with an example. The second part of the chapter explained several extensions of this basic heuristic method with suggestions for further improvements.

Chapter 7

The Genetic Algorithm

"A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, and die gallantly.

Specialization is for insects."

Robert A. Heinlein

7.1 Introduction

A genetic algorithm is a general purpose parameter optimization technique. A genetic algorithm is composed of four main elements: the genetic representation (encoding), the genetic engine, the genetic operators and the fitness function. For completeness, in the following sections, each of these components will be explained. Of course, the details refer to the final version of the genetic algorithm to be used in the experimental studies that follow.

7.2 Genetic Representation

In this research, the genetic chromosome representation is based on the work as reported in [Leich 95], although earlier work of marker-based coding style exists [Fullmer & Miikkulainen 92, Moriarty & Miikkulainen 93]. This representation resembles very closely the way in which real DNA chromosomes are represented. In particular, a chromosome is a sequence of single-valued cells, each of which is termed a locus and represents the basic information building block. Within a chromosome certain sub-sequences of loci represent genes, each of which denotes an instruction that is used, in conjunction with zero or more other genes, to build a fuzzy rule. Genes are separated by specific sequences of loci that act as the delimiting boundaries, or START-END marks. Such a mark is unique whether it represents a start or an end depends on the context. Any locus between genes are considered junk (called introns in biological literature) and are therefore ignored. These junk loci or non-coding DNA blocks are called introns. Note that those junk sequences may nonetheless be helpful since they lessen the disruptive effect of genetic operators.

This representation allows the expression of a variable number of rules in a chromosome and that of a variable number of antecedent variables in each rule. Overall, each chromosome represents an entire collection of instructions needed to construct a fuzzy rule set, that is, the representation is an *indirect encoding*. Each population therefore consists of sets of emerging translated rulesets. For GAs implementing either individual or group strategy (see section 5.6) the chromosomes do not encode consequents (as all of them share the same consequent). However, for use of the global strategy the consequents are also encoded.

Each chromosome there represents a variable number of antecedent blocks (and the consequent if needed). Such blocks are composed of the definitions of a label and a varying number of hedges that are applied to modify the label. To represent situations where hedges are absent a specific term of *no-hedge* is introduced. These

definitions are simply denoted by cardinal numbers. The chromosome is composed of series of zeroes, ones, and two special tokens that respectively stand for START/END of a definition and a NEW RULE. These tokens can be positioned in any locus. To avoid disruption when mutation generate happens to generate the symbol of NEW RULE, only two consecutive NEW RULEs will treated as the end of the definition of a rule. Any single NEW RULE will be ignored. As mentioned above, for the individual or group strategy it is intended that the consequent of all the generated descriptive rules to be the same as that of the original approximative rule, therefore there is no need of such output classes being encoded. For the global strategy the last integer number read before the END marker is considered to be the consequent, encoding the corresponding class. If the end of the chromosome is reached with a partial definition of a rule then such partial definition is ignored. This means that the incomplete part of the chromosome from the last (double) NEW RULE marker until the end of the chromosome is treated as a junk (intron). The algorithm that interprets a given chromosome to form descriptive rules can be summarized as follows:

- For hedges definition, repeat the following *max_hedges* times (one per possible hedge), where *max_hedges* stands for the number of hedges allowed to be used to modify a given fuzzy set:
  - Read up to find START/END
  - Read the minimum number of zeroes and ones to define a hedge (ignoring START/END)
  - Keep reading up zeroes and ones until the next START/END
  - Obtain the code number for the hedge from the clean sequence of zeros and ones read
- Label definition, do the following once
  - Read up to find START/END

  - Read the minimum number of zeroes and ones to define a set (ignoring START/END)
  - Keep reading up zeroes and ones until the next START/END
  - Obtain the code number for the set from the clean sequence of zeros and ones read

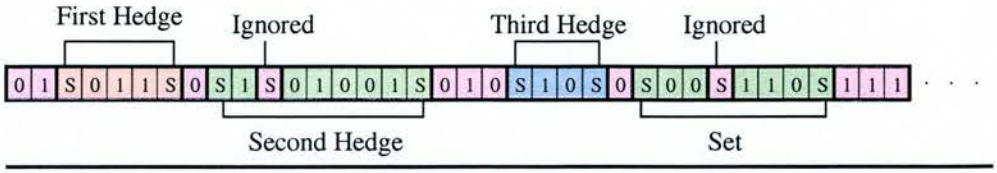
- If NEW RULE is found at any point then form a rule antecedent using the complete read blocks. Discard any partially read block and start interpreting the definition of a new rule.

- If the global strategy is being encoded use the last read number as the consequent, if this number was part of a completed block discard the block.

According to [Leich 95] this representation is very robust against the disruption effect (of destroying an emerging rule) that genetic operators usually have. A considerable proportion of a chromosome does not represent useful information and can be considered as introns, but these introns allow the abortion of such disruptions and absorb mutations (decreasing its effective rate and acting as a self-regulator). The genetic recombination may fall in such portions, and this does not alter those definitions potentially already optimized. The approach allows for a variable length of the rules to be interpreted and for independence of the crossover points.

An example should help to understand this representation. Suppose that three different hedges named as hedge number 1, 2 and 3 plus the no-hedge (indication no hedge being applied) and 20 different descriptive sets may be used. (These 20 sets are all that are defined on all variables.) Thus, two bits are enough to define the hedges and five bits are sufficient to define the descriptive sets. The strings given in figures 7.1 and 7.2 , with each "S" indicating a START/END marker can therefore be decoded as follows.

Suppose that three hedges are encoded. The process looks for an "S" marker to start a definition so the first two numbers are ignored and the first hedge starts to be



		Modulo	Scaling
First Hedge	011	3	1
Second Hedge	101001	1	2
Third Hedge	10	2	2
Set	00110	6	3

Figure 7.1: Example of GA codification using three hedges

defined from the fourth digit (the zero). The definition ends in the second "S" giving the number 011 which represents the first hedge. This number can be decoded to a particular hedge in two ways. One is by taking the modulus of the number of hedges available. In this case it will be hedge coded as 3 ($[3]_4 = 3$ with $[.]_N$ representing the modulo N function and $N = 4$ here). Another way is to scale the range using the upper bound to obtain an integer identifying which hedge the number represents. In this case the number contains 3 bits. With 3 bits 8 combinations can be generated but there are only 4 possible hedges (including the no-hedge), the scaling factor is thus $\frac{1}{2}$. So the hedge will be interpreted as one coded as number two ($3 * \frac{1}{2} = 1.5$, leading to an upper round of 2) if this method is used.

Having identified this hedge, the process will search for another "S" so the next zero is ignored. Once found it starts to read numbers but, as at least two of them are needed the "S" after the first one is therefore ignored since only one number was read and the process keeps reading until an "S" is again found. The number for this new hedge to be identified (ignoring the in-between "S") is 101001=41. Using the modulo method hedge number one is obtained. Using the scaling method the scaling factor is 4/64=1/16 and the hedge would be the one named number 3.

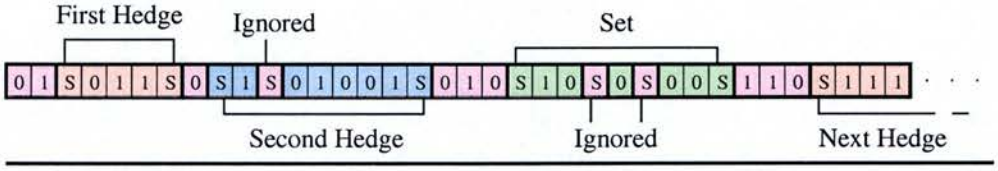
The process is repeated for the third hedge in the chromosome. In this case the

number is 10 and, as the result, both methods interpret it as the predefined hedge coded as number two.

Now a set is to be decoded. As five numbers are needed the "S" at the third locus inside the definition is ignored, returning 00110 as the result. Using the modulo decoding this number represents the sixth fuzzy set. Using the scaling decoding the scaling factor is 20/32 that multiplied by 6 is 3.75, the interpretation is therefore the fourth descriptive set.

Thus, the given string, using three hedges, can be translated as:

- H3H1H2S6, using modulo decoding; or
- H2H3H2S4, using the scaling method.



		Modulo	Scaling
First Hedge	011	3	2
Second Hedge	101001	1	3
Set	10000	16	11

Figure 7.2: Example of GA codification using two hedges only

The same string example, but decoding two hedges only, is shown in figure 7.2. Following the described procedure, using two hedges, the string can be translated as:

- H3H1S16, using modulo decoding; or
- H2H3S10, using the scaling method.

7.3 Genetic Engine

The genetic algorithm adopted here is a steady-state one [Whitley 89], that is, instead of generating a whole population that replaces the previous one (known as

a generational GA [Goldberg 89]), only one or two children are generated each cycle and they replace one or two members of the population depending on the replacing schema chosen. One parent is selected by linear ranking and the other parent is selected by random choice. Each child replaces a random member of the worst half of the population. The search stops when the best half of the population does not improve for a prescribed number of generations or a fixed number of evaluations. To avoid premature convergence, a minimum number of generations is enforced and a high mutation rate is set. Diversity is maintained thanks to the random replacement within the worst half of the population.

The population is sorted by fitness value and divided into two halves. The better half is used to measure similarity between members of the population to check if there has been any improvement in the population. When a new member is created by crossover and/or mutation it will replace a random member of the worse half of the population, no matter whether the new member is better or not. This mode of replacement keeps diversity in the population while, in the meantime, avoids disruption of the best members which are already in the better half.

The above means that all new children join the population but it does not mean that they are considered as successful. A new member is considered successful when it is in the better half of the population after the sorting procedure ends. Successful children will reward the operators that create them. More about these operator rewards will be discussed in section 7.5.

7.4 Genetic Operators

Four different mutation and four different crossover operators have been implemented in order to investigate what combinations may lead to a good translation. The mutation rate and the rate at which a different crossover is used are both allowed to change

dynamically. The mutation rate increases when the population members become very similar.

As crossover always takes place (the GA is a steady-state), a special no-crossover operator is included, to allow mutations to be applied without a crossover. To decide among the different crossovers/mutations a biased random schema is used: Each different crossover/mutation has a fixed minimum chance of being applied. If applying a certain crossover/mutation has so far resulted in a successful new member then its chance to be applied in future generations is increased at the expense of the rest. Similarly, if it is unsuccessful its chance for future use will decrease. In general, during the GA execution there will be more unsuccessful tries than successful ones. Hence, the rate at which a particular crossover/mutation is applied should decrease more slowly than the rate of increase of applying a successful crossover/mutation. Empirically, the inclusion of this dynamic schema helps improve significantly the performance of the GA employed. More information about this dynamic crossover/mutation rate is given in section 7.5.

7.4.1 Flipping loci

This mutation changes the contents of random loci in the chromosome. In particular there is a fixed 5% chance per locus of being mutated. If the locus contains a 0, 1 or START/END then it is changed to another symbol with the following probabilities: 25% of being START/END, 37% of being 0, 37% of being 1 and 1% of being NEW RULE. Even if chosen, if the locus to be mutated contains a NEW RULE symbol then 90% of the times it will be left as it is, unmodified, due to the rather drastic consequences of mutating such symbol (therefore having an effective probability of being modified of 0.1%). These values for the probabilities of changing the symbols were obtained after preliminary tests. They seem to be very intuitive. The different chances for 1's, 0's and START/END reflects the different proportions of such symbols in the chromosome.

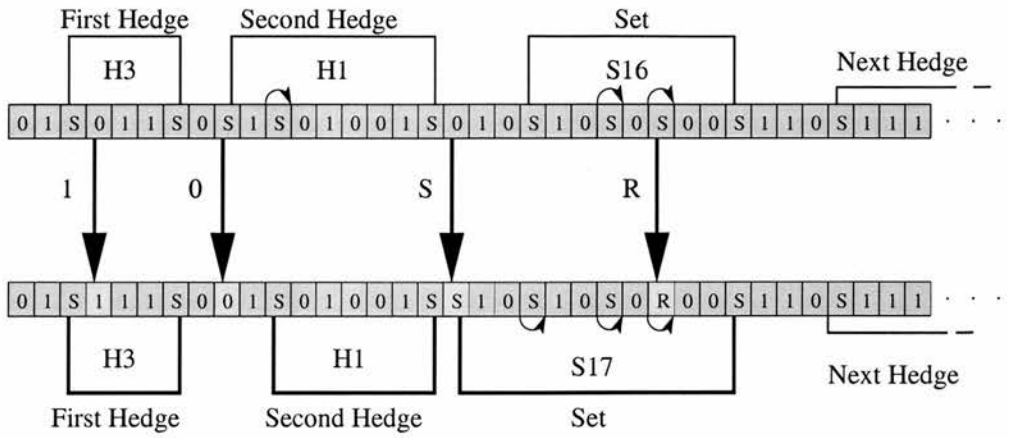


Figure 7.3: Example of Flipping Loci Mutation

An example of such mutation can be seen in figure 7.3. Certain values of the loci are modified and, as a result, the interpretation of the Set varies from 16 to 17 (small arrows indicate ignored locus). It can be seen that the changes do not always modify the final interpretation of the chromosome, showing the robustness of the representation against changes in its basic structure. The introduction of a single NEW RULE symbol does not modify the interpretation at all.

In summary, this is a low level mutation that modifies the basic instructions given to build the whole descriptive ruleset.

7.4.2 Flipping hedges/sets mutation

This mutation works by changing certain randomly identified sets and/or hedges in the antecedent of the emerging rule. The chromosome is decoded and then, depending if the component is a hedge or a set, it undergoes mutation with a certain probability, namely 2.5% if it is a hedge and 1% if it is a set. The new value is a random value among the possible ones for the component using a uniform distribution. Finally, the section of the chromosome that defines such a modified component (hedge or set) is rewritten so it encodes the new value assigned to the component.

This is performed by generating a bit string of a length between the minimum bits needed to encode the hedge (or set) up to two times that length. Then a sequence of zeroes, ones and some ignored START/END is created in a way that, when decoded, represents the desired number to be rewritten in the chromosome.

The different chances for the hedges are a result of the assumption that it is more interesting to change the combination of hedges (and thus the shape of the centroids that represent the sets) than the sets themselves (that is, the main location of the centroids).

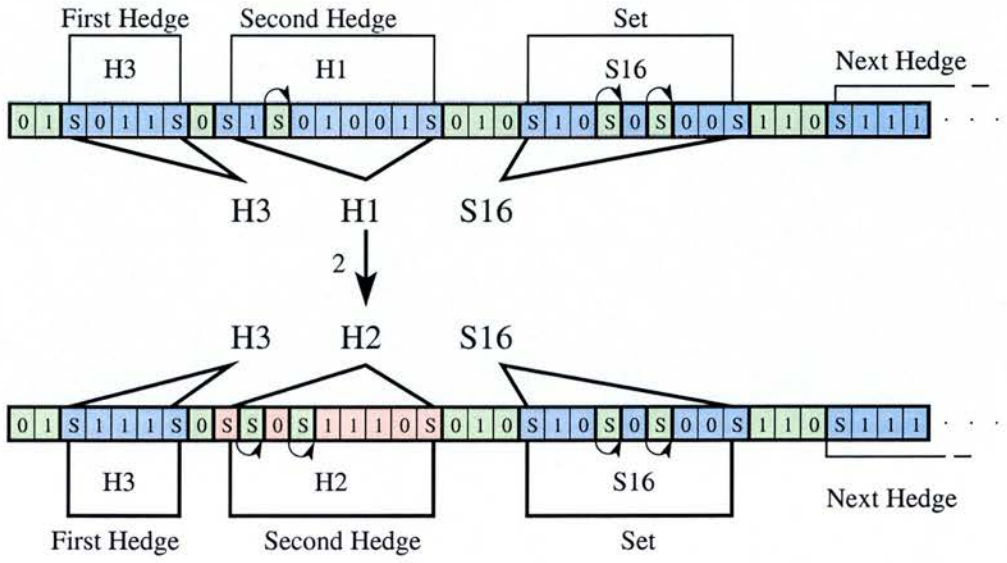


Figure 7.4: Example of Flipping Hedges/Sets Mutation

Figure 7.4 shows an example of this mutation. The chromosome is decoded (using modulo) into the sequence H3H1S16. Then the second hedge undergoes a mutation, changing to the value 2. The section of the chromosome that defines this hedge is then completely rewritten in a way that it is decoded as the selected value. In this case the sequence is S0S1110; removing the ignored S symbols produces the binary number 01110, and using modulo 4 (the number of possible hedges) gives the value 2.

In summary, this mutation changes a whole sequence that defines a particular component of the chromosome. In this sense, it is considered as a high level mutation.

7.4.3 Adding a rule mutation

This mutation adds a rule to the emerging fuzzy rule set by inserting into the chromosome a new rule instruction sequence. The new rule so created is generated randomly as follows:

1. A number of antecedent conditions is first selected. The number of such conditions is a random number with a bell shaped probability distribution with the center equal to half the number of the variables involved in the problem. This way many of the newly created rules will have a reduced number of variables in the antecedent, pushing the GA to produce rules easier to read and, at the same time, reducing the dimensionality of the problem (if possible).

2. Once the number of conditions is obtained the set and the hedges for each condition are selected. Sets are chosen using uniformly distributed random numbers, but for the hedges half of them will be the no-hedge. Again, this is done to push the GA to produce clearer rules.

3. If the translation procedure uses the global strategy a consequent is generated using the proportion of the examples of a particular class to chose the output class.

Once a rule is completely created the corresponding sequence of symbols that defines the rule is inserted in a random place between two rules. Although for the emerging rules of a chromosome the location within the chromosome itself is irrelevant, its place can affect when other genetic operators are applied and therefore its own potential descendants.

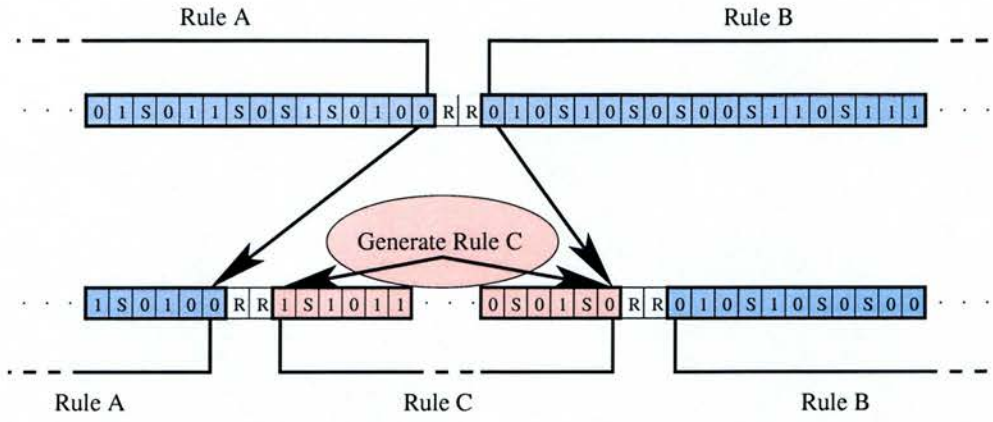


Figure 7.5: Example of Adding a rule Mutation

Figure 7.5 shows an example of the application of Adding a Rule mutation to a chromosome. As the changes take place at the level of the rules definitions this mutation can also be considered as a high level mutation.

7.4.4 Removing a rule mutation

This mutation removes a rule from the emerging fuzzy rule set by removing from the chromosome a whole sequence that defines a rule. The rule to remove is chosen randomly with a uniform distribution.

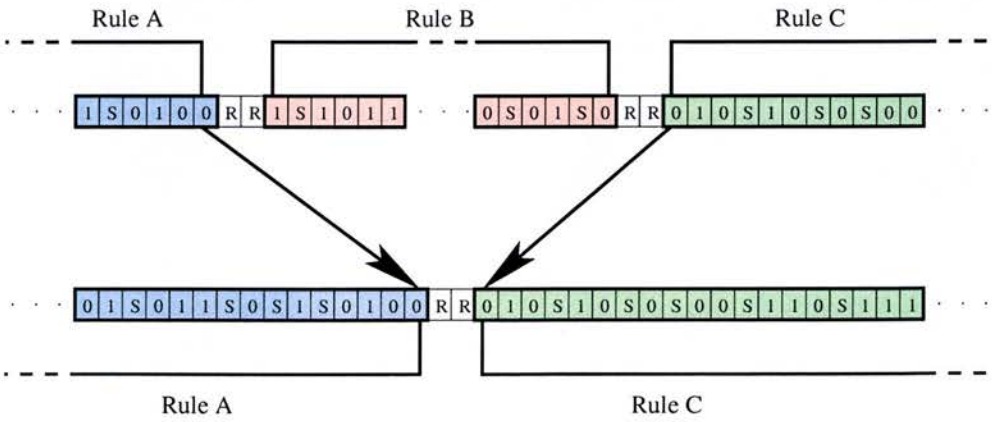


Figure 7.6: Example of Remove a Rule Mutation

In figure 7.6 one example of this mutation is shown. The rule termed B is chosen to be removed. The operator just cuts in the chromosome sequence that defines this rule and discards it.

Again, the changes occur at the level of rule definitions, i.e. it makes sense at the level of what is represented, therefore this mutation is another high level mutation.

7.4.5 Rulemixing crossover

This crossover uses two parents and generates two children. It goes through both parents rule by rule and copies the current rule to one of the children with a given probability. In particular each rule of the first parent has 90% chance of being copied to the first child and 10% to the second. Likewise, each rule of the second parent has 10% chance to go to the first child and 90% chance to go to the second child. In this way the children are similar to the parents but with 10% of its rules interchanged.

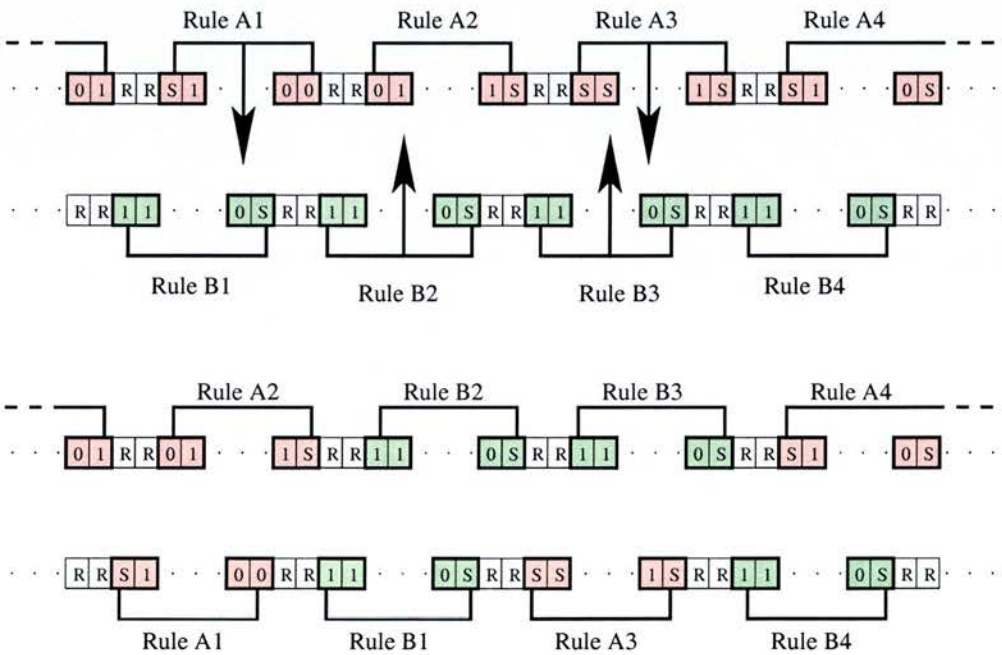


Figure 7.7: Example of Rulemixing crossover

In figure 7.7 an example of this mutation is shown. The sequence in which the

parents (termed here Parent A and Parent B) went through is: A1, B1, A2, B2, A3, B3, A4 and B4. The rules A1 and A3 are selected for interchange in parent A and the rules B2 and B3 are also selected for interchange in parent B. Therefore, A1 goes to child 2, B1 goes to child 2, A2 goes to child 1, B2 goes to child 1, A3 goes to child 2, B3 goes to child 1, A4 goes to child 1 and B4 goes to child 2. The final sequence for child 1 is A2, B2, B3, A4 while that for child 2 is A1, B1, A3 and B4.

In summary, this operator shuffles the rules of two rule sets by cutting the two parent chromosomes into chunks of individual rules (individual genes) and rearranging randomly these chunks, biased to interchange only 10% of the material, into two new chromosomes. It is also a high level operator.

7.4.6 One point crossover

This operator is classical. It works by choosing a point in each of the chromosomes of the two parents and generating a child using as a copy of the first part of the first parent from the beginning of the chromosome up to the point of interchange, and a copy of the second part of the second parent from the point of interchange up to the end of the chromosome. Another child is generated in a similar way but using the first chunk of the second parent and the last chunk of the first parent. The points of interchange are chosen randomly using a uniform distribution.

In this classical operator the point of interchange is typically the same on both chromosomes. This is so because usually all the chromosomes have the same length and, if the point is the same on both parents, the children will have also the same length. However, in this work, such constraint is obviously not necessary. Therefore, any point of each parent can be chosen. The length of the children can be, and in fact usually is, quite different than that of their parents. There may be cases where a child can get most of the chromosomes of the parents and the other inherit almost no chromosome (the case of a point being close to the beginning of a parent and the other being close to the end of the other parent), but this is not a problem as the extreme children will,

almost certainly, have a poor fitness. Nevertheless, the total length of the children will be the same as that of the parents.

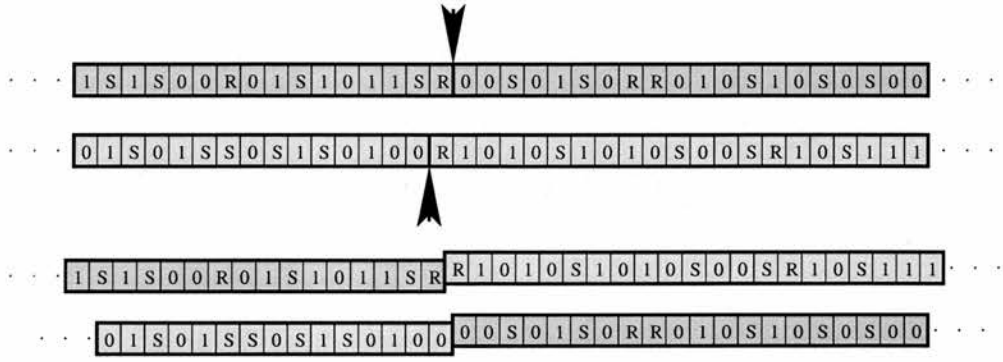


Figure 7.8: Example of one point crossover

Figure 7.8 shows an example of this operator in action. The interchange points are marked with the arrows and the chromosomes are colored to ease the interpretation of which parts of the parents are in which child.

In summary, this classical one point operator cuts the parent chromosomes into two pieces in random loci and then swaps the resulting halves. This is a low level operator as no chromosome contents are interpreted.

7.4.7 Two point crossover

This operator is also a classical. The difference between this operator and the one point operator is that here two points of interchange are chosen instead of one. In this case, the pieces delimited by the two points and exchanged between parents to generate the children.

As with the last operator, the classical two point crossover usually uses the same two points for both parents and maintains the length of the child chromosomes. Here, again, such constraint is not necessary and each parent may choose the interchange points independently. The points are chosen using a uniform distribution.

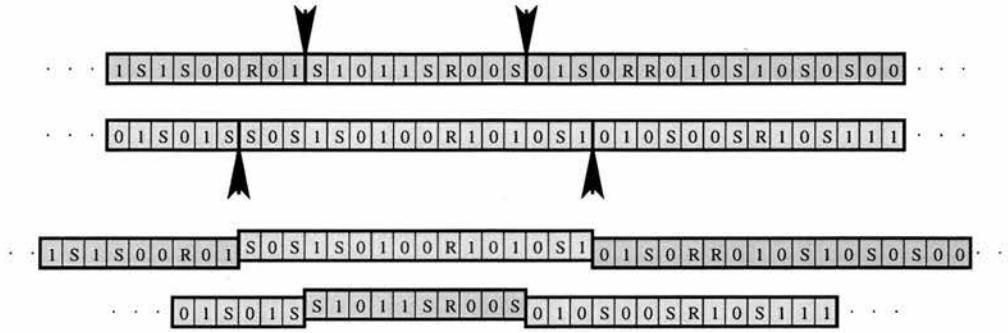


Figure 7.9: Example of two point crossover

Figure 7.9 shows an example of this crossover operator in use. Clearly, the two pieces are of a different length.

In summary, this two point crossover operator is similar to the one point crossover in cutting the parent chromosomes but this one interchanges the sections of the chromosomes contained between two random cutting points. Again, this is a low level operator as it does not consider, as the basic elements, the sequences that define a component (a hedge, a set, etc.).

7.4.8 Uniform crossover

This operator is the third of conventional crossover operators. It is a logical extension of the two points crossover: Each pair of locus in the same position are given the chance to be exchanged in the children, that is, the first child gets the locus of the second parent and the second child gets it from the first parent. Of course, if they are not exchanged then the first child gets the locus of the first parent and the second child gets it from the second parent.

As with the previous two classical crossovers, the uniform crossover was originally designed to work on fixed sized chromosomes. As the current representation has a chromosome of variable size there is not such a concept as "locus in the same position". Therefore, the operator works in a slightly different way. Without losing generality it

can be assumed that one of the parents is longer than the other. A random number is chosen between zero and the difference of lengths between the chromosomes. The locus that is in the long parent in the position marked with that random number will be considered as the first locus and will be paired with the first locus of the short chromosome. Thereafter all following loci are paired likewise . Now the operator can be applied. Both extra pieces of the long chromosome (from the beginning up to the random point and all the remaining unpaired loci) join the first child at its beginning and the end if the long is the first parent, otherwise join the second child.

It is easy to see that this operator is rather disruptive so the chance to be swapped between parents is fixed to be 5% in implementation.

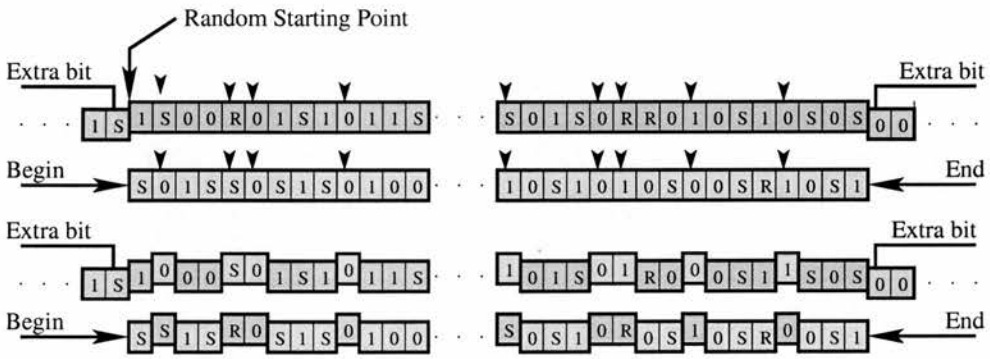


Figure 7.10: Example of uniform crossover

Figure 7.10 shows an example of this operator, with the first parent longer than the second. Clearly, there are extra pieces at the beginning and at the end of the first chromosome that will not participate in the exchange (those extra bits are shown in green in the figure) and they will remain part of the first child. From the initial random point there is a 5% chance that the loci will be exchanged.

In summary, the uniform crossover interchanges the loci of two chromosomes randomly with the number of loci to be interchanged being about 5% of the number of loci of the shorter parent chromosome.

7.4.9 Cover of the uncovered

This is one of the two special operators that are introduced to tackle some deficiencies in the search that have been detected within this research. In a strict sense it is a high level mutation and similar to "add a rule" mutation. However, instead of adding a random rule a heuristically constructed one is introduced in the chromosome. In particular it creates a rule that covers some training data that the rest of the rules in the chromosome do not cover.

This mechanism works by selecting an example of the data that is not covered by any rule and building a rule that covers it. The rule is obtained in a similar way to that of the famous Wang & Mendel algorithm. It builds descriptive fuzzy rules from examples [Wang & Mendel 92]. It takes on the example to cover and searches, for each variable of the problem, which is the set (without any hedge) that covers the example's coordinate in that variable with a highest membership value. Then, it combines all these sets into a rule using the AND operator, with the output value being that of the example. The reason for not using hedges is because it dramatically increases the number of potential candidates to become a chosen set. In fact, this increment is one of the facts that make Wang & Mendel algorithm not very practical (together with the explosion in the number of rules [Mendel 01]). Having created a rule, the position in the chromosome to insert it is randomly chosen.

In figure 7.11 an example of this procedure is depicted. It shows a two dimensional problem with two classes. Examples of one class are shown as $\times$'s and those of the other class are represented with +'s. The figure shows two rules, the bluish one has as output class $\times$, the greenish has as output the class +. The different shades of blue or green represent the different firing strengths. The sequence HHS inside the sets that define the rules shows that such sets are combinations of two hedges plus an original set. Notice that, in the left figure, there are three uncovered points (surrounded by red circles). The example chosen to be covered is indicated with an arrow. On the right

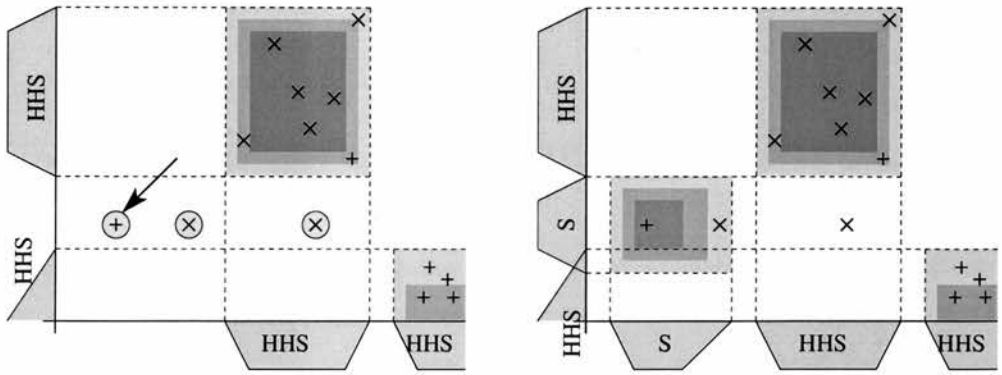


Figure 7.11: Example of Special Operator Cover Uncovered Points

hand a new rule with single original sets (no hedges) is created so it maximizes the strength of the rule in the indicated place. The rule is green as the example chosen to be covered is of the class +. Note that some examples of the other class happen to be covered by the new rule. This is left to the GA to improve further later using other operators in future generations.

In summary this special operator adds a new rule to the chromosome so that it covers some examples (at least one) that were not previously covered. This is also a high level operator.

7.4.10 Split rule

This is the other specially introduced operator, in an effort to promote specialization in rules that show a high error. This operator reflects the intuition that by splitting a rule into three is hoped that some of these newly created rules get most of the errors and would be eventually eliminated later by other operators.

The operator works by finding the rule in the chromosome that has the greatest ratio of incorrect versus fired examples (with the latter being correctly and incorrectly covered examples). Of course, the examples are those classified by that very rule. Once such a rule is selected one of its antecedents parts (a set with possibly some

hedges) is randomly chosen. Then the rule is replaced by three copies of itself. These three copies are modified such that the difference between the three new rules rests in the outmost hedge of the chosen condition: One will have the hedge LOWER as the outmost hedge, another will have the hedge MID, and the last will have the UPPER hedge. The consequent of the rule is the same as that of the original. The rest of the rules also remain unchanged.

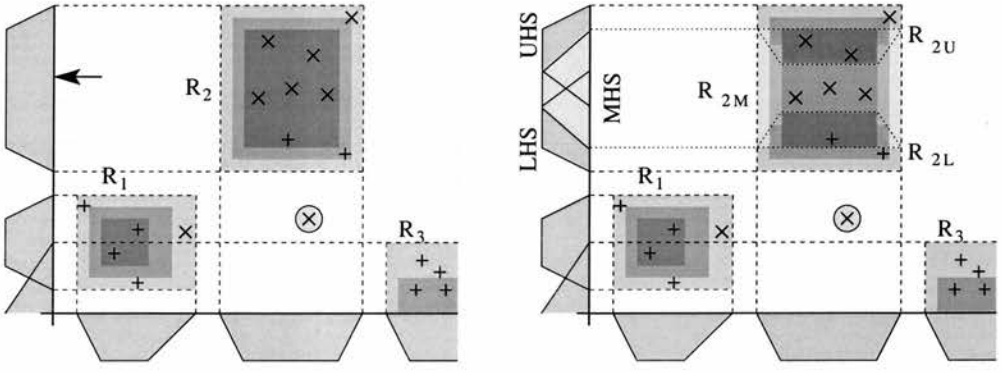


Figure 7.12: Example of Special Operator Split Rule

Figure 7.12 shows the process described. In this case rule R_2 (the bluish one) is chosen to be changed because of the high proportion of misclassified +'s (remember that blue is associated with class $\times$ and green with class +). The proportion of incorrect versus fired examples is $\frac{2}{6}$ for this rule (while the R_1 has $\frac{1}{5}$ and R_3 has $\frac{0}{4}$). The component of the antecedent chosen to be modified is marked with an arrow. The right hand figure shows that the component now is changed into three. One set has the form LHS, i.e LOWER plus (any) HEDGE plus the (original) SET, another set has the form MHS (MID-HEDGE-SET), and the last is of the form UHS (UPPER-HEDGE-SET). Thus, the original rule has been multiplied to become three rules. The originally covered space is now divided into three parts. In this engineered example rule R_{2L} covered all the incorrect cases. It is now the work of the GA to remove this rule in its later generations. Rules R_{2M} and R_{2U} are now perfect rules that hopefully will remain in the genetic pool for a long time.

In summary, this last operator tries to modify rules with a high proportion of incorrectly classified examples in an attempt to reduce their error. It creates three rules using the hedges LOWER, MID and UPPER introduced in this work. It is also a high level operator.

7.5 Dynamic Operator Probabilities

With so many different operators available to the algorithm it is a very difficult (if not impossible) task to decide the optimal probabilities of applying each of them. Some of the operators produce quite long jumps in the search space while others are more fine grain ones. It is a good practice to promote the former type of operators at the beginning of the search so as to avoid local minima and to decrease their importance in the latter stages of GA running. Likewise fine grain operators are typically more appropriate for the final moments of the search to produce a hill climbing effect (as the local minima are likely to have been avoided). Of course, these are more a rule of thumb than theory.

Hyper-heuristics [Fang *et al.* 94, Hart & Ross 98, Hart *et al.* 98, Ross *et al.* 02] is starting to appear as a robust approach to tackle, among others, this problem of when to use what operators. It is a term coined to describe the idea of using a number of different heuristics (in this case operators) together, so that the actual heuristic applied may differ at each decision point. Hyper-heuristics are, in essence, heuristics to *choose* heuristics. It is important not to confuse hyper-heuristics with the widely used term meta-heuristics, as this latter term refers to heuristics that control simpler heuristics for a rather narrow range of problems while hyper-heuristics choose among a wide and diverse set of full heuristics to solve the widest range of problems possible. The approach works by learning which operators must be applied to solve a problem given the current state of its solving.

However, in this work, to partially lessen the problem a dynamic schema of

assignment of probabilities is developed. The procedure is simple in nature. Starting from equal probabilities it promotes the operators that generate fitter children and penalizes those that generate poor children. In this way it self-adapts to the particular stage of the search and to the special characteristics that a particular problem may have. Of course the schema has its own parameters to be adjusted such as the amount of reward to give to a successful child and the rate at which unfit descendants are penalized. Nevertheless, the whole process is substantially simplified.

In particular, to decide among the different crossovers/mutations the following biased random schema is used. Each crossover has a fixed minimum chance of being applied (with a total of 50% shared among all possible crossovers). If applying a certain crossover has so far resulted in a successful new member then its chance to be applied in future generations is increased at the expense of the rest. In implementation for each successful child the probability of being chosen for that operator increases in a step of 5%. Similarly, if it is unsuccessful its chance for future use will decrease. In general, during the execution of a GA there will be more unsuccessful tries than successful ones. Hence, the rate at which a particular crossover is applied decreases more slowly than the rate of increase of applying a successful crossover. In implementation, this decreasing rate for unsuccessful insertion of new children is set to 0.02%, of which is given to the rest. Empirically, the inclusion of this dynamic schema helps improve significantly the performance of the GA employed.

Figure 7.13 shows a typical evolution of the probabilities of crossover operators (including the two special operators). Each line corresponds to one of them. It is interesting to note that there are several operators that play a predominating role in certain stages of the search. In the cases depicted there is one operator that seems not to help much in this particular search. A brief study was performed to detect if there are operators that always predominate over others and if there are others that perform poorly all the time. Unfortunately, such a study gave no significant conclusions. It seems that the performance of each operator is dependent on the problem and also

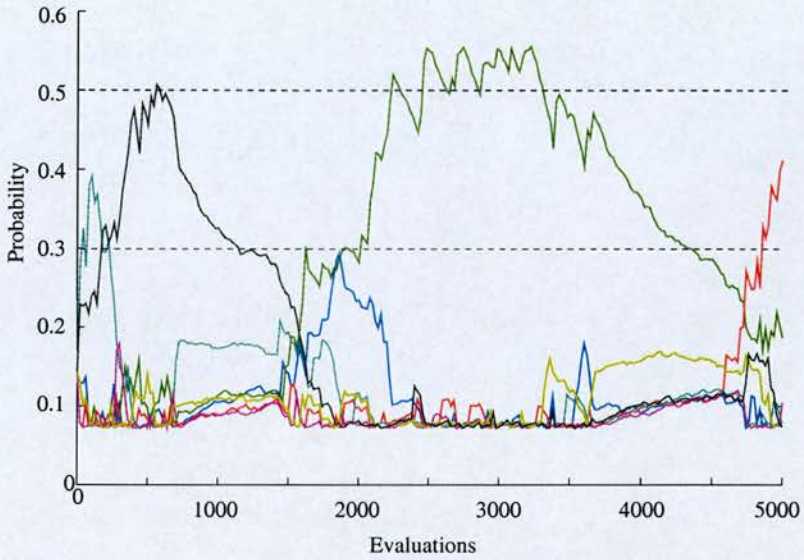


Figure 7.13: Xover and special operators probability evolution

partially on its initial luck. This, of course, reflects the nature of GA-based search in general.

Some operators seem to perform a similar kind of search in a number of problems because they alternate in dominance at the same stage, depending on the initial seed. This suggests that if these "coupled" operators could be chosen at the stage in which they start to be useful they soon dominate (as they get more chances to be chosen) at the expense of the rest that would do a similar job. This is more a conjecture than a proved fact. The study of the reasons of such behavior falls far beyond the scope and interest of this thesis (more focused in the translation process itself). The important conclusion is that all the explained operators may become important at certain stages for the tested problems (see chapter 8) and therefore they are all included. Incidentally, this brief study helped drop an operator which is a variation of the uniform crossover at high level that showed very low performance (possibly due to poor parameter optimization).

Figure 7.14 also shows typical evolutions of the probabilities, but this time of the mutation operators. As with the crossover operators no mutation consistently

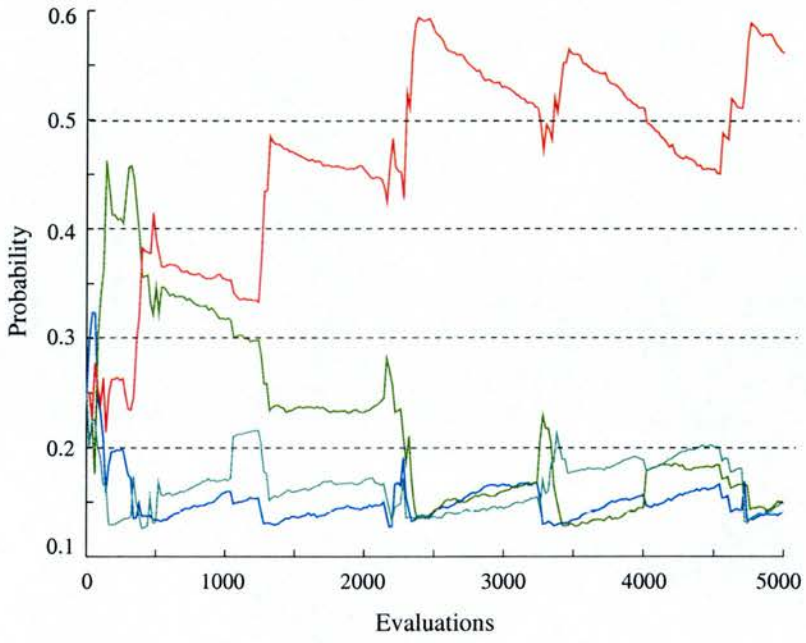


Figure 7.14: Mutation probability evolution

outperformed the others and again which operator to choose seems to be problem dependent. In the particular case depicted in the graph one of the mutations happens to dominate most of the run.

7.6 Fitness Function

GAs can be fairly insensitive to the precise choice of fitness function [Rana *et al.* 96]. Of course, a good evaluation will help the system to search in the proper direction, but the main purpose of evaluation functions is to help the selection of the parents. Therefore, it is the selection mechanism that can have a more dramatic effect on the outcome of the algorithm. Selection controls and defines the transformation from an evaluation to the fitness assigned to a particular chromosome.

Theoretical work has been carried out in the literature, in order to study the relation between a fitness function (in the evaluation sense) and the difficulty of a GA to solve

a problem guided by it. It is established that fitness functions have a direct relation with the shape of the search landscape [Jones 95, Kallel *et al.* 01]. As it assigns the "altitude" of a point in the search space an evaluation function should try to generate smooth hills or mountains instead of plateaus with sharp towers. The fact that the landscape is also shaped by the representation chosen and that, in most cases, there are several objectives to achieve at the same time, makes the creation of good evaluation functions very difficult for most complex modelling problems.

One of the most important features that an evaluation function should have is speed. Everytime a new chromosome is created, or the environment changes in a way that should be measured (like, for example, a change in the weights of the objectives), one or more evaluation functions must be executed. In this work, the speed of the execution of the evaluation function depends on a) the length of the chromosome (as it has to be decoded), b) the number of rules that are contained within the chromosome and c) the number of training data points. The latter two are possibly the most important factors. To speed up the calculation of firing strength minor optimizations are included. In particular, the rules are stored in a tree like structure and also as soon as a fire strength of one is found then the corresponding rule is chosen to be returned. Regarding the number of training points, potential optimizations are outlined in section 9.2.3.

As indicated in chapter 2, there exist in the GA literature several approaches to deal with problems that have multiple objectives, including the aggregation approach [Goldberg 89], Non-Pareto approach [Schaffer & Grefenstette 85] and Pareto-Based approach [Goldberg 89]. The present work adopts the first of these, as it offers a conceptually simpler method by converting multiple objectives into a compounded single objective. In particular the aggregation function used is the Sum of Weighted Global Ratios (SWGR) [Bentley & Wakefield 96, Bentley 99]. The aggregation method first independently normalizes each objective with respect to the best and worst value ever found for it and, then, weights and adds together each objective to form the single overall fitness value. Whenever a new maximum or minimum value is found for

an objective the whole population fitness is recalculated. To avoid recalculation of the different objectives the non-normalized values of such objectives are stored with the chromosome so only the normalization step is performed. The aggregated objectives used in this work are summarized in table 7.1. The fitness function expression is:

$$f = w_1 \cdot o_1 + w_2 \cdot o_2 + \cdots + w_8 \cdot o_8 \tag{7.1}$$

with w_i being the weight for each objective and o_i the normalized value of each objective. If $\sum w_i = 1$ the fitness function value will be normalized also.

Table 7.1: Summary of the aggregate objectives.

Error of type 1 training data
Error of type 2 training data
Error of type 3 training data
Number of rules
Overall error variance
Number of Correctly Classified (%)
Number of Incorrectly Classified (%)
Number of Not Covered (%)

Figure 7.15 shows the evolution of the fitness in one of the executions, in particular a conversion of rules defined on the Iris problem (more information about this problem will be given in chapter 8). The blue line represents the fitness of the best element of the population. The green line represents the average fitness of the better half of the population, the group of protected members. The red shows the average of the total population. The oscillations in the first evaluations are due to the re-scaling that are produced whenever a new best or worst value for an objective is found. After some generations it is unlikely that these extreme values have to be updated and the

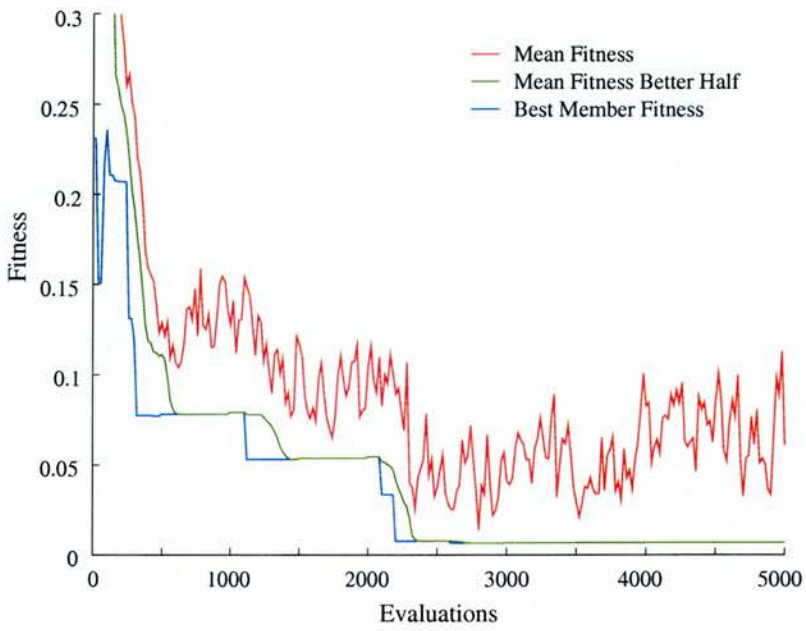


Figure 7.15: A typical fitness evolution

oscillations disappear. Note, while considering the shape of the fitness graph, that a gradual change is maintained from functional equivalence to classification objectives.

The diversity is kept thanks to the replacement schema devised. The ragged red line shows this point as, at least, there exists one chromosome (possibly more) that is performing differently than the best half of the population (pushing the red line toward worse fitness).

The selection pressure in the better half of the population is moderate. The speed at which the green line catches up with the blue shows the generations needed for the better half of the population to converge and gives an idea of this selection pressure.

The increases in fitness are rather steep and sudden, suggesting that the fitness function is highly non-linear. This is something to be expected by the very nature of the descriptive fuzzy partition of space. There is no smooth transition nor a way to establish an ordered relation among the combinations of hedge-sets. Therefore if the

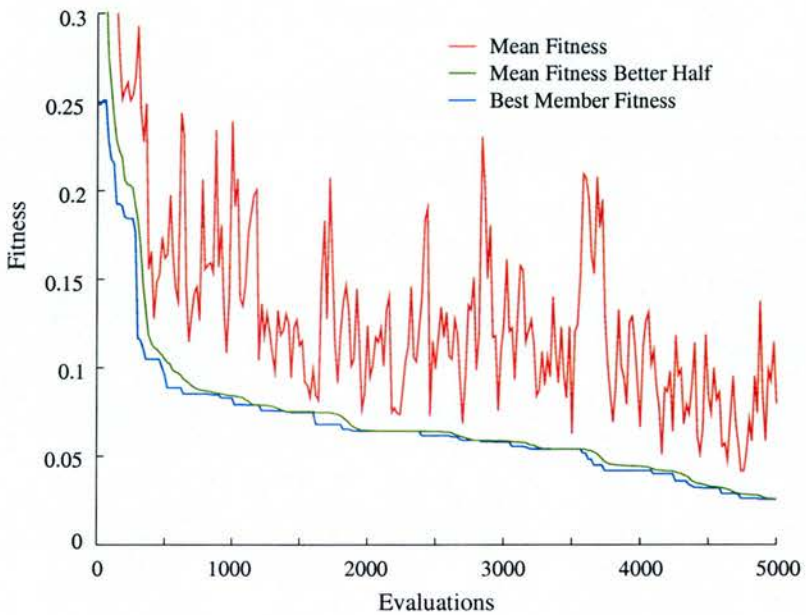


Figure 7.16: Another typical fitness evolution

GA is performing a hill-climbing-like work the increases will jump significantly. If the GA is doing more than a random-sampling-like work the jumps are also to be expected.

Another example of the evolution of the fitness function values of this algorithm can also be seen in figure 7.16 (with respect to a Diabetes Problem translation; see chapter 8 for the description of this problem). This problem is bigger and more difficult to translate than the Iris one. The jumps are, as expected, accordingly smoothed. Nevertheless, improvement is clearly obtained with the proposed GA.

7.7 Summary

This chapter has described the specific genetic algorithm devised for use in the present research. The description was divided with respect to the main components of a genetic algorithm. All the genetic operators were explained with graphic presentation of the procedures. A section was devoted to the mechanism that is responsible for self-adaptive assignment of the proportions of the different GA operators.

The above genetic algorithm can be improved in many ways. However, it is not the main aim of this research to obtain a perfect genetic algorithm for the application, but to show that the translation procedure from approximative to descriptive models can be done without substantial loss of accuracy. It was for this reason that once the modelling results had reached a good accuracy no extra work was directed toward the genetic algorithm improvements. Fortunately GAs are very forgiving algorithms, even if they are badly implemented, or poorly applied, they will often still be able to produce acceptable results [Davis 91]. Tuning and testing even more alternative variants of the algorithm (with its correspondent programming and debugging) is a tedious and time consuming task that remains as an important piece of future work.

Finally, as a final remainder of the most important features of the GA used to implement the translation a summary of them is provided in table 7.7.

Table 7.2: GA Features

- Marker-based representation
- Variable length chromosome
- Indirect representation
- Steady-state
- Pittsburgh style
- Use of introns
- Low epistasis
- Selection: 1st Parent Linear Rank, 2nd parent Random
- 1 No xover operator: Allows mutation only
- 4 different crossover (high and low level)
- 4 different mutations (high and low level)
- 2 special operators (high level)
- Dynamic crossover/mutation probability assignment
- Replacing random among half the worst (always).
- Stop when maximum generations consumed
- Aggregation fitness function (SWGR)
- Multi-objective (8 objectives)

Chapter 8

Results on Benchmark Problems

"Example isn't another way to teach, it is the only way to teach."

Albert Einstein

"There are three kinds of lies: lies, damned lies, and statistics."

Benjamin Disraeli

8.1 Introduction

This section presents computer simulation results of applying the proposed descriptive modeling techniques to a number of benchmark problems. The experimental background is first described. A simple example in terms of resultant descriptive rules is given next, in comparison to the original approximative rules. Comprehensive results are then reported and analyzed, supported by comparisons with related work.

8.2 Experimental Premises

Good benchmarking is a necessary but often neglected aspect of this kind of study. Many researchers appear to fail in performing proper benchmarking for their new

algorithms/methods/approaches. A poor statistical analysis or data preparation renders the best tables and graphics useless and any conclusion taken on those results untrustworthy.

The most common problems include:

- Insufficient number of problems to test the performance.
- Impossibility of reproducing the test.
- Using unknown or non available problems not allowing comparison with other known algorithms/methods/approaches.
- Using strange/not common measures of error or fitness preventing also comparison.

More information about the above is given in [Pretzel 95] and [Prechelt 96]. The discussion below is given with a hope to be as objective as possible, via not avoiding all these issues.

It is worth noting that many researchers feel the temptation to present in their work only well behaved data. That does not means that the problems presented were easy to solve, but that the ones they use to sustain their conclusions are the ones that fit them best [Prechelt 96]. In this work results reported are not chosen due to any particular well behaved problem. Examples of different, though all well known problems are shown and typical graphs are likewise presented. As the aim is to compare between the old descriptive modeling techniques and the present research the obsession for accuracy is softened and so is the temptation to present the best results.

Although, in general, Fuzzy Modeling is a very powerful tool to work on datasets with noise or very complex non-linear relationships, sometimes it can not overcome serious deficiencies of several data sets. Points to take into account include, for example, the proportions of the classes in the set. If it is very unevenly distributed

the systems tend to mostly classify the large classes, sometimes even to extremes of ignoring tiny classes. These are revealed in some of the experimental results to follow.

Unless stated otherwise the statistics presented in this chapter regarding the experiments using the GA have been obtained using the following procedure:

1. Run the GA 100 times using different seeds.

2. Bootstrap 1000 samples from the 100 real runs. If r runs of the GA were allowed for each translation, and the best of these r runs were to be taken as the final translation, then each bootstrapped sample is obtained as the best of r random samples among the 100 real runs.

3. The mean error of the classification and the standard error of the bootstrapped sample $\left(\frac{\sigma}{\sqrt{n}}\right)$, with σ being the standard deviation and n the sample size, are calculated.

4. Confidence intervals are calculated as 2.58 times the standard error, this gives a 99% confidence for the mean (assuming a normal distribution).

5. Tables are built showing the mean and the confidence interval.

8.3 The Benchmarks Used

To demonstrate the proposed approach at work, benchmark classification problems are used here [UCI], including the Breast Cancer, Diabetes, New Thyroid, Wine, and Iris datasets. Table 8.1 summarizes the set-ups of these datasets.

The following subsections will describe these problems. A description of each data set is given along with the origin of the data, its size and some properties it has. All problems have been previously used in literature, although possibly the training and test sets may be different. Some basic statistics for each problem are also included.

Table 8.1: Classification problems

Name	No. of Inputs	No. of Output Classes	No. of Samples
Breast Cancer	9	2	683
Diabetes	8	2	768
Iris	4	3	150
New Thyroid	5	3	215
Wine	13	3	178

Parts of the problem explanations are directly extracted from the Proben1 Documentation Files. The problems selected share some common features:

- All inputs are continuous.
- All outputs are discrete.
- All problems are classification problems.
- Data is split into two sets, with rules extracted using the training set and tested using the test set.

Except for the Iris Problem, where an initial expert fuzzy specification was available, the rest of problems have been tested using a random fuzzy specification.

8.3.1 Iris Problem

This is R.A. Fisher's famous iris data, published in [Fisher 36] and very widely used as a test for statistical analysis techniques. The data set includes measurements of the sepal length, sepal width, petal length and petal width on 50 specimens from each of three species: (1) Iris setosa, (2) Iris versicolor, (3) Iris virginica. Table 8.2 shows the proportions of each species and over both training and test sets, as well as in the ANFIS resultant rules (those to use as the given approximative rules for translation).

Table 8.2: Iris Problem Data and Rule proportions.

	Number	(1) Setosa	(2) Versicolor	(3) Virginica
Training Samples	112	33.03%	33.92%	33.03%
Test Samples	38	34.21%	31.57%	34.21%
ANFIS Rules	4	1	1	2

This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. One class is linearly separable from the other 2; the latter are *not* linearly separable from each other. This is an exceedingly simple domain, however.

8.3.2 Breast Cancer Problem

This dataset was created based on the "breast cancer Wisconsin" problem dataset from the UCI repository of machine learning databases. The data was originally obtained from the University of Wisconsin Hospitals, Madison, from Dr. William H. Wolberg [Wolberg & Mangasarian 90].

The data set concerns the problem of diagnosing Breast Cancer, trying to classify a tumor as either benign or malignant based on cell descriptions gathered by microscopic examination. Input attributes are, for instance, the clump thickness, the uniformity of cell size and cell shape, the amount of marginal adhesion, and the frequency of bare nuclei. Table 8.3 presents the proportions of each class for each set and in the approximative rules.

This problem has 9 input attributes and 1 output attribute with 699 examples. The output attribute is one of two classes indicating whether benign or malignant. A total of 65.5% of the examples are benign in this data set.

Table 8.3: Cancer Problem Data and Rule proportions.

	Number	Benign	Malignant
Training Samples	512	63.67%	36.32%
Test Samples	171	69%	30.99%
ANFIS Rules	18	4	14

8.3.3 Diabetes Problem

This dataset was created based on the "Pime Indians diabetes" problem dataset from the UCI repository of machine learning databases.

The problem addressed is to decide whether a Pime Indian individual is diabetic or not, based on personal data (age, number of times pregnant) and the results of medical examinations (e.g. blood pressure, body mass index, result of glucose tolerance test, etc.). Table 8.4 gives proportions of each class for each set and in the approximative rules produced by ANFIS.

Table 8.4: Diabetes Problem Data and Rule proportions.

	Number	Negative	Positive
Training Samples	576	65.1%	34.89%
Test Samples	192	65.1%	34.89%
ANFIS Rules	28	16	12

This problem has 8 input attributes and 1 output (2 classes) attribute with 768 examples. A total of 65.1% of the samples are diabetes negative. Although there are no missing values in this dataset according to its documentation, there are several senseless 0 values. These most probably indicate missing data. Nevertheless, such data are treated as real input, thereby introducing some errors (or noise) into the dataset.

8.3.4 Wine Problem

This dataset is available at the UCI repository of machine learning databases. The data was originally obtained from the Institute of Pharmaceutical and Food Analysis and Technologies of Genoa, Italy in [Forina].

The data set shows the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The input attributes include data as alcohol, Malic acid, Magnesium, etc. Table 8.5 lists the proportions of each class for each set and in the approximative rules.

Table 8.5: Wine Problem Data and Rule proportions.

	Number	Cultivar 1	Cultivar 2	Cultivar 3
Samples Training	133	33.08%	39.84%	27.06%
Samples Test	45	33.33%	40%	26.66%
ANFIS Rules	6	3	2	1

This problem has 13 input attributes and 1 output attribute with examples. The output attribute is the cultivar where the wine grapes where collected. A total of 33.1% of the examples are of the first type, 39.8% of the second, and 26.9% of the third.

8.3.5 New Thyroid Problem

This dataset is also available at the UCI repository of machine learning databases. The data was originally obtained from Danny Coomans, Department of Mathematics and Statistics, James Cook University, Townsville, Australia.

The data set reflects the problem involved in the diagnosis of irregular function of the thyroid gland. Five laboratory tests are used to try to predict whether the thyroid of a patient belongs to the class euthyroidism (normal), hypothyroidism or hyperthyroidism. The diagnosis is based on a complete medical record, including

anamnesis, scan etc. The input attributes include information about the percentage of the T3-resin uptake, the total serum triodothyronine and the basal thyroid-stimulating hormone (TSH). Table 8.6 presents the proportions of each class for both training and test sets and in the approximative ruleset produced by ANFIS.

Table 8.6: Thyroid Data and Rule proportions.

	Number	Normal	Hyper	Hypo
Training Samples	161	69.56%	16.14%	14.28%
Test Samples	54	70.37%	16.66%	12.96%
ANFIS Rules	5	3	1	1

More specifically, the problem involves 5 input attributes and 1 output attribute with 215 examples. A total of 69.7% of the examples are normal, 16.2% are hyperthyroidisic and 13.9% belong to the hypothyroidism class.

8.4 Experiments for Redefined Trapezoidal Hedges

To independently test the performance of the conventional and proposed implementations of hedges the translation mechanism described in this thesis was used to carry out descriptive data-driven modeling (though any other descriptive modeling technique may be employed as an alternative for this purpose). This first set of experiments is oriented just to show the differences that the implementation of hedges can make. In the following sections another set of experiments, this time aiming to measure the quality of the proposed descriptive fuzzy rule generation, will demonstrate that the translation mechanism outperforms existing descriptive techniques.

8.4.1 Set-up for redefined trapezoidal hedges

As indicated previously, ANFIS was used to obtain the original approximative fuzzy models, one per problem. Also, to ensure the readability of the resulting descriptive models, it is disallowed for any consecutive use of more than two linguistic hedges over a given fuzzy set within the experiments.

Although, in theory, the translation process works by having a heuristic translation to serve as the generator of the initial population for the GA, to avoid any possible advantage by taking this initial heuristic population (that has been found to produce better results when used as empirically shown, see later) random rules are used to initialize the GA. To minimize the possibility that a particular random ruleset may potentially benefit one or another group of hedges ten different random rulesets have been generated, and GA runs were started using one of these rulesets at random.

The version of the GA used has no special features designed for the proposed new hedges. Detailed parameters of the GA were set the same across all experiments and so no advantage may be taken by the present work. The figures to be presented below will show the mean error of the translated rulesets depending on the number of runs allowed for the GA.

8.4.2 Results for redefined trapezoidal hedges

Table 8.7 lists the results of using the original ANFIS models and table 8.8 those of applying the descriptive models obtained by employing conventional linguistic hedges. Comparing to these results, table 8.9 lists the outcome of applying the hedges defined in this work. "Rules" in these tables stands for the number or average number of rules produced while "Size of Rules" stands for the average number of conditions in the antecedent of the rule. Note that some figures include confidence intervals as the values are obtained from statistics from a sample. To provide a fair comparison between the use of new hedges and that of the traditional ones (which lack the novel

detailisation hedges and therefore may be argued of having less variety), experiments without involving the three new hedges *LOWER*, *MID* and *UPPER* were also carried out and their results are reported in table 8.10.

Table 8.7: Classification error and number of rules using the ANFIS model

Problem	ANFIS		
	Train Error (%)	Test Error (%)	Rules
Breast Cancer	0.4 %	4.1 %	18
Diabetes	15.7 %	26.6 %	28
Iris	0.8 %	2.6 %	4
New Thyroid	3.1 %	1.8 %	4
Wine	0 %	2.2 %	6

Table 8.8: Results, with 99% confidence intervals, of employing traditional hedges

Problem	1 run of the GA			
	Train Error (%)	Test Error (%)	Rules	Size of Rules
Breast Cancer	3.91 % $\pm$ 0.21	6.94 % $\pm$ 0.16	6.94 $\pm$ 0.15	3.59 $\pm$ 0.07
Diabetes	31.38 % $\pm$ 0.28	31.78 % $\pm$ 0.43	7.55 $\pm$ 0.13	2.39 $\pm$ 0.05
Iris	14.24 % $\pm$ 0.47	15.24 % $\pm$ 0.64	3.57 $\pm$ 0.04	2.47 $\pm$ 0.03
New Thyroid	17.84 % $\pm$ 1.05	17.95 % $\pm$ 1.18	4.16 $\pm$ 0.05	3.52 $\pm$ 0.07
Wine	15.20 % $\pm$ 0.37	20.61 % $\pm$ 0.49	7.76 $\pm$ 0.12	2.78 $\pm$ 0.05
Problem	Best out of 5 runs of the GA			
Breast Cancer	2.09 % $\pm$ 0.04	5.97 % $\pm$ 0.06	7.98 $\pm$ 0.12	4.04 $\pm$ 0.06
Diabetes	27.67 % $\pm$ 0.09	27.66 % $\pm$ 0.26	7.97 $\pm$ 0.13	2.20 $\pm$ 0.05
Iris	8.08 % $\pm$ 0.26	7.24 % $\pm$ 0.37	3.55 $\pm$ 0.04	2.50 $\pm$ 0.03
New Thyroid	9.61 % $\pm$ 0.09	10.80 % $\pm$ 0.18	4.24 $\pm$ 0.03	3.46 $\pm$ 0.06
Wine	10.24 % $\pm$ 0.37	16.93 % $\pm$ 0.40	8.53 $\pm$ 0.11	2.99 $\pm$ 0.04

When comparing the classification results of a descriptive model with those of the

Table 8.9: Results, with 99% confidence intervals, of using all types of new hedge

	1 run of the GA			
	Train Error (%)	Test Error (%)	Rules	Size of Rules
Cancer	2.54 % ± 0.06	6.63 % ± 0.07	9.06 ± 0.17	3.22 ± 0.07
Diabetes	28.50 % ± 0.31	25.74 % ± 0.36	9.82 ± 0.23	1.99 ± 0.04
Iris	3.64 % ± 0.10	3.56 % ± 0.15	4.36 ± 0.06	2.56 ± 0.04
Thyroid	6.24 % ± 0.20	9.57 % ± 0.34	6.30 ± 0.11	3.30 ± 0.06
Wine	6.61 % ± 0.29	8.84 % ± 0.41	7.44 ± 0.11	2.77 ± 0.05
	Best out of 5 runs of the GA			
Cancer	1.71 % ± 0.02	6.25 % ± 0.06	8.15 ± 0.18	3.36 ± 0.07
Diabetes	24.78 % ± 0.10	22.65 % ± 0.16	9.57 ± 0.19	1.96 ± 0.04
Iris	2.30 % ± 0.05	3.27 % ± 0.15	4.30 ± 0.05	3.01 ± 0.04
Thyroid	3.73 % ± 0.08	6.20 % ± 0.25	6.22 ± 0.11	3.30 ± 0.07
Wine	3.44 % ± 0.10	6.49 % ± 0.31	7.22 ± 0.10	3.03 ± 0.05

original approximative model obtained by ANFIS, the performance of any descriptive model does not appear to be impressive. However, this is not the point of this investigation; much better descriptive models can be acquired when an initial heuristic translation is employed. The intention of the present experiments is to show the differences between applications of different sets of hedges.

The results leave little doubt about the clear advantage of the use of the redefined hedges. Even better performance is reached when the newly introduced hedges are exploited. Either way, the use of hedges implemented in the present work significantly outperforms the use of traditional ones. This result is obtained consistently across all tested problems. The performance improvement is particularly dramatic for the last three problem cases (namely, Iris, New Thyroid and Wine). In addition to these results, figure 8.1 shows in greater detail the evolution of the error measured as the number of

Table 8.10: Results, with 99% confidence intervals, of using new hedges but excluding detailisation ones

Problem	1 run of the GA			
	Train Error (%)	Test Error (%)	Rules	Size of Rules
Cancer	2.43 % ± 0.06	6.80 % ± 0.09	8.77 ± 0.15	3.91 ± 0.07
Diabetes	29.04 % ± 0.35	26.78 % ± 0.46	9.26 ± 0.20	2.59 ± 0.06
Iris	4.16 % ± 0.13	4.83 % ± 0.16	4.48 ± 0.04	2.53 ± 0.03
Thyroid	7.58 % ± 0.21	13.14 % ± 0.36	5.91 ± 0.07	3.64 ± 0.06
Wine	7.89 % ± 0.30	10.75 % ± 0.40	7.87 ± 0.14	2.25 ± 0.04
	Best out of 5 runs of the GA			
Cancer	1.79 % ± 0.23	6.45 % ± 0.07	9.19 ± 0.11	3.91 ± 0.06
Diabetes	25.82 % ± 0.08	23.79 % ± 0.18	8.49 ± 0.16	2.73 ± 0.06
Iris	2.58 % ± 0.05	5.48 % ± 0.15	3.93 ± 0.04	2.60 ± 0.03
Thyroid	5.18 % ± 0.10	10.69 % ± 0.31	5.99 ± 0.07	3.75 ± 0.07
Wine	4.24 % ± 0.11	8.67 % ± 0.34	8.11 ± 0.15	2.32 ± 0.04

GA runs allowed for getting the best model is increased. This highlights further the success of the present research.

8.5 Set-up for experiments on Translation

Again, a particular version of ANFIS, which uses bell-shaped approximative sets, and which has been optimized, was trained for this set of experimental studies. The resulting approximative ruleset is employed as the original set of rules for translation for each problem case outlined in section 8.3.

As indicated before, the output of the heuristic method is used to act as the generator of the initial population for the GA that performs finer search for the final descriptive rules. The heuristic method used was the extended 2, allowing two nodes per layer with both I- and S-thresholds set to zero. This provides a good compromise

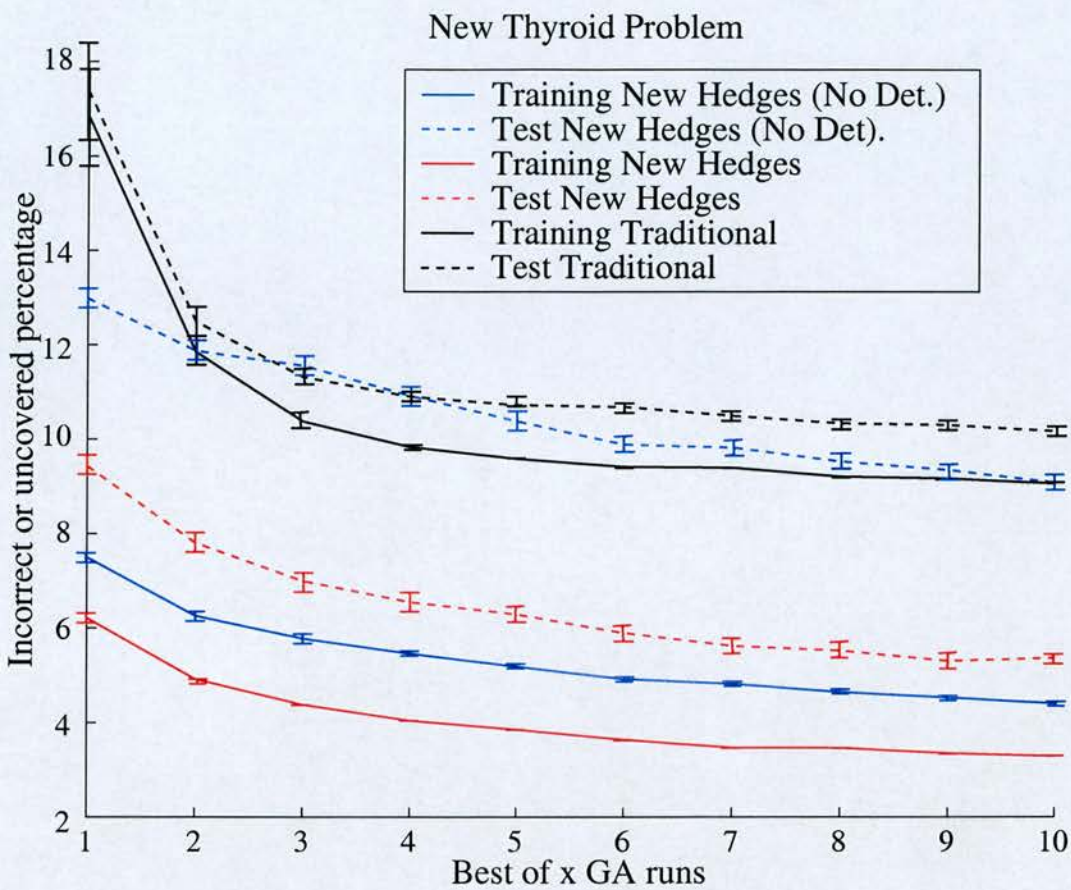


Figure 8.1: Classification error for New Thyroid problem using the best of several GA runs. Error bars show the 99% confidence interval where applicable.

between accuracy and complexity. To ensure the readability and understandability of the resulting descriptive ruleset, the maximum number of hedges (including the *NOT* operator) allowed to be applied to a given set is limited to 2.

For comparison, the pure descriptive induction algorithm as outlined in section 3.5, which is a form of exhaustive search with different parameter settings is also tested. In particular, this algorithm (referred to as Lozowski's algorithm) has a parameter that trades off between the model accuracy and the size of generated ruleset. It determines the minimum difference between the firing strengths of any potential rules that have the same antecedent but different class values. In the present investigation, this parameter

is set up with reference to the number of rules that the heuristic method has generated to ease comparison. That is, several possible values of the parameter were tried for each problem, and the resultant model with the number of rules closer to that generated by using the Extended Heuristic 2 is chosen as representative for the Lozowski's algorithm.

Also, for comparison purposes, results obtained by running the standard C4.5 algorithm [Quinlan 93] are included. To be simple and fair for comparison, for each problem considered the fuzzification was carried out proportionally with respect to the size of the universe of discourse of the individual variables. That is, for each variable, the distance between its maximum and minimum value within the data set is divided such that all of them approximately cover an equal range of the underlying real values, with soft boundaries of course. The fuzzy sets resulting from such a partition are regarded as the given descriptive sets. This is implemented for illustrative purposes and is not necessary in practical applications of the present work. In fact, the whole idea is that the fuzzy partitioning and labelling will be done by user/experts. To demonstrate the effectiveness of this approach the fuzzification scheme used has only 3 labels per variable. Figure 8.2 shows such an example for the New Thyroid problem.

The GA uses a population of 30 rulesets and run for 10000 evaluations (not generations). Note that, to allow comparison, for a given execution the evaluations are divided among the different subproblems or sub-GAs, if applicable. These sub-GAs are GAs running on parts of the translation. For example, for the individual translation strategy a sub-GA is a GA used to translate a particular rule. In group strategy it is a GA running a particular class. The term sub-GA does not apply to the global translation strategy as there is only one problem, i.e. the translation of the whole approximative ruleset. Thus an execution of the global strategy will have 10000 evaluations, while a group execution for 3 classes will have 3333 evaluations per class translation and an execution of the individual translation strategy for 10 rules will have 1000 evaluations for each rule translation.

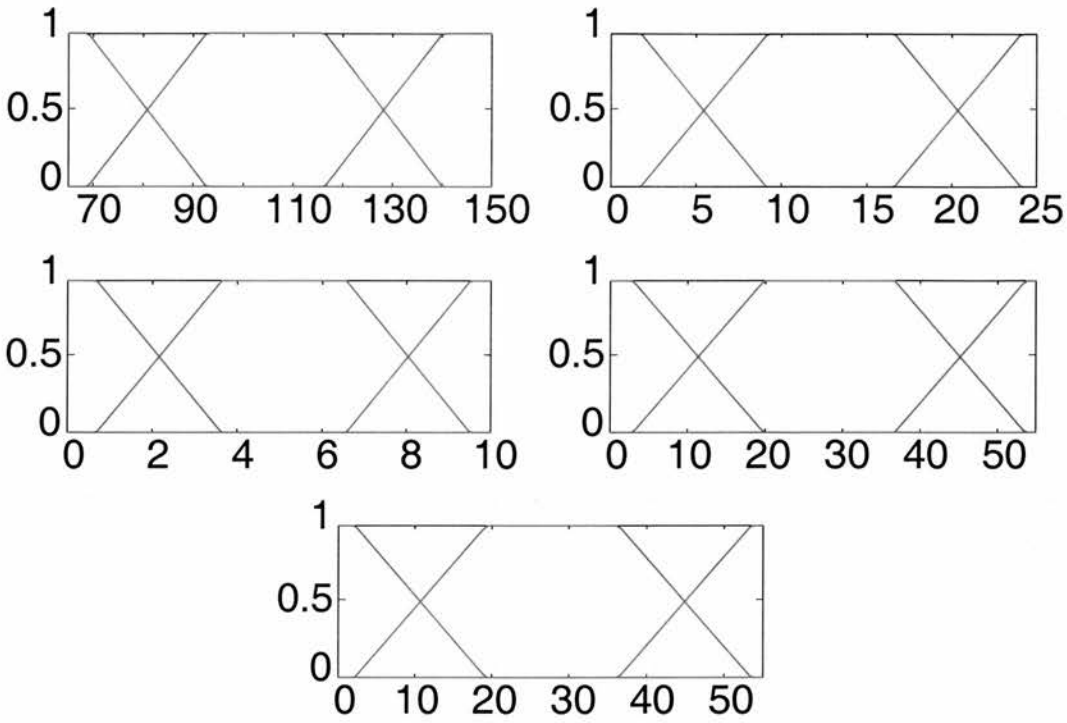


Figure 8.2: Descriptive sets for New Thyroid

As GA execution is computationally affordable, it is worthy to execute the genetic search for as many times as possible. This is in order to obtain the best among as large a number of different translations as possible to act as the final translation. The figures to be presented below will show the mean error of translated rulesets depending on the number of runs allowed for the GA. Such error measures are obtained using a bootstrapping of 1000 samples over 100 real runs. Experimental results are given for the GA guided by *functional equivalence* and also for the GA guided by classification (error) rate alone. To avoid possible overfitting each dataset has been separated into a training set containing 75% of all the given data and a test set comprising the remaining 25%.

8.6 Example of Transparency Gained by Translation

> "Few things are harder to put up with than the annoyance of a good example"
>
> *Mark Twain*

To reflect the fundamental differences between approximative and descriptive modeling, an example ANFIS ruleset for the Iris problem [Fisher 36] and one of its descriptive translations are given here. The approximative rules are:

Rule A1:

if x_0 is $Bell_{0.70,1.99,5.00}$ and x_1 is $Bell_{0.36,2.06,3.60}$ and
x_2 is $Bell_{0.31,2.11,1.35}$ and x_3 is $Bell_{0.40,2.04,0.03}$ then *Class* is *Setosa*

Rule A2:

if x_0 is $Bell_{1.02,1.99,6.23}$ and x_1 is $Bell_{0301,2.31,2.84}$ and
x_2 is $Bell_{0.39,2.18,4.02}$ and x_3 is $Bell_{0.11,2.31,1.53}$ then *Class* is *Versicolor*

Rule A3:

if x_0 is $Bell_{0.58,1.99,7.53}$ and x_1 is $Bell_{0.71,2.00,3.06}$ and
x_2 is $Bell_{0.27,2.11,6.33}$ and x_3 is $Bell_{0.64,1.99,2.18}$ then *Class* is *Virginica*

Rule A4:

if x_0 is $Bell_{0.14,1.91,6.25}$ and x_1 is $Bell_{0.53,2.15,2.46}$ and
x_2 is $Bell_{0.24,2.11,5.31}$ and x_3 is $Bell_{0.37,1.97,2.30}$ then *Class* is *Virginica*

Obviously, these rules are hardly readable, though they may be generated very rapidly and they may well generalize the given training data.

Suppose that the labels attached to a variable's three possible descriptive sets are named long, medium and short or thin, medium and wide, depending on whether the variable refers to length or width respectively. Each descriptive set may be modified

by zero up to two hedges (as defined in section 4). Given the approximative rules, the following translated rules were generated:

```
Rule D1:
  IF Petal Width IS Upper Thin THEN Iris-Setosa
Rule D2:
  IF Sepal Length IS Medium AND Sepal Width IS Greatly Medium AND
     Petal Length IS Very Medium AND Petal Width IS Very Medium
  THEN Iris-Versicolor
Rule D3:
  IF Petal Width IS Lower Greatly Wide THEN Iris-Virginica
```

To examine the translation results, figure 8.3 shows the descriptive partition used in this example. Figures 8.4 and 8.5 plot the membership functions involved in the antecedent of the approximative rule A1 and its translation D1 (only the Petal Width has a value, Upper Thin). Also, figures 8.6 and 8.7 plot rule A2 and its translation D2.

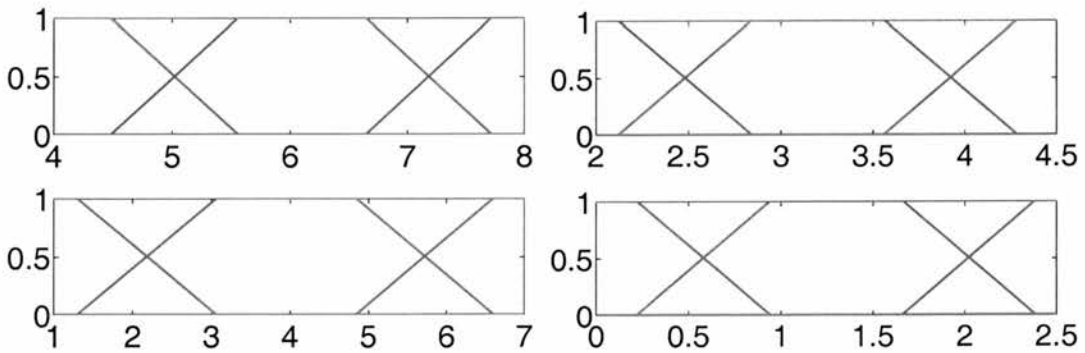


Figure 8.3: Descriptive sets for sepal length and width, and petal length and width, respectively (Iris)

These descriptive rules may appear rather different from the original approximative ones, yet they have the same functionality. The translated rules are represented in linguistic words with predefined meanings. In using such a rule base, both the

interpretation of the inferences performed and the explanation of the fuzzy system itself becomes straightforward. Very interestingly, for this example, the number of resultant descriptive rules is actually less than that of their original, whilst these two rulesets entail the same classification accuracy. Also, two out of the three descriptive rules are more concisely represented than any of the four original rules.

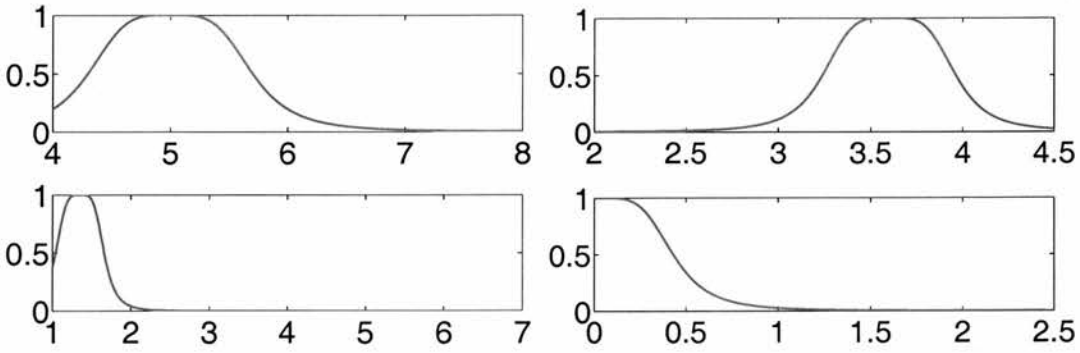


Figure 8.4: Fuzzy sets for approximative rule A1 (Iris Setosa)

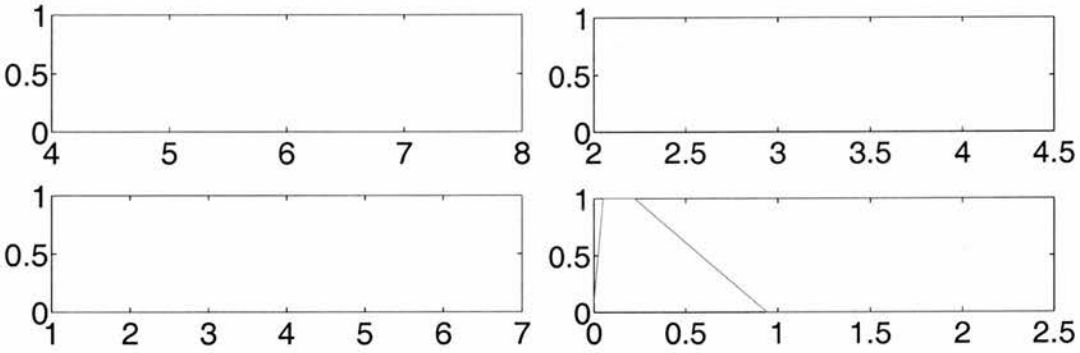


Figure 8.5: Hedged sets for descriptive rule D1 (Iris Setosa)

It is worth recalling that, in general, there can be multiple descriptive rule sets that may each become the translation of a given approximative rule set. Even descriptive rule sets that translate a given approximative rule set with a very similar accuracy may look rather different. In general, it is very difficult to expect a double translation (which

starts from a descriptive model, produces a dataset from this model, generates an approximative model of that dataset and then translates it back to a descriptive model) to reproduce the same original descriptive ruleset or even to resemble the original closely. This is more obvious in modeling complex systems where many possible combinations of rules can yield similar rule-firing results. It is sufficient if a resultant descriptive model is accurate (which it will be using the current work, provided that the original approximative model is itself accurate) *and* is directly interpretable.

Note that no principled way exists to guarantee that a data driven rule induction mechanism would reproduce exactly the same rules that were first used to create the data. What can be expected most is to be able to generate a set of rules that match the data as closely as possible, hoping that such a set of rules do not differ too much from the underlying one. Although multiple descriptive rulesets may be obtained from one given approximative model, only one optimized is eventually chosen to act as the translation. Thus explanation will be unique for a problem at hand once the translation process is completed.

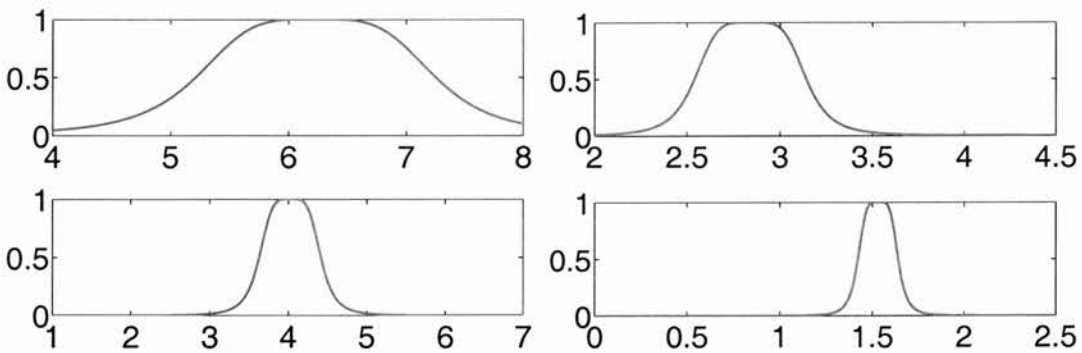


Figure 8.6: Fuzzy sets for approximative rule A2 (Iris Versicolor)

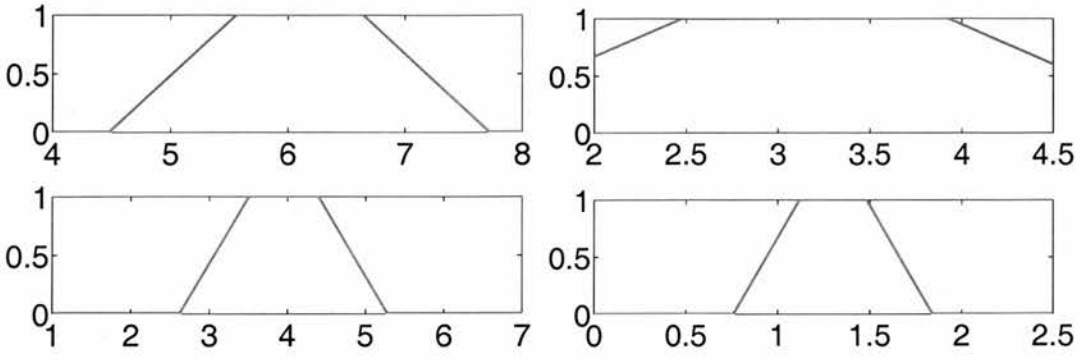


Figure 8.7: Hedged sets for descriptive rule D2 (Iris Versicolor)

8.7 Results on the Accuracy of translated models

This section presents and discusses the results regarding the classification accuracy of the learned fuzzy ruleset.

8.7.1 Effects of translation strategies

First of all, it is interesting to investigate the effects of using different translation strategies. Figure 8.8 collects the results of such experiments on the Diabetes problem, as an example, whilst results on other problems are very similar. It can be seen that the group strategy gave better results than the global and individual ones during the training phase. In testing, all the three strategies performed very similarly overall.

However, difference exists in terms of the number of rules needed to achieve the similar test results. Individual translation gave more, sometimes many more, rules as the GA tried to produce good translations locally. The pressure on the search exerted by the number of rules objective on individual strategy was not so high as the pressure by the same objective in carrying out group or global translations, where more data points were involved. When using group or global strategies it may happen that several approximative rules can be covered by just one descriptive rule, with potential savings in the total number of rules generated.

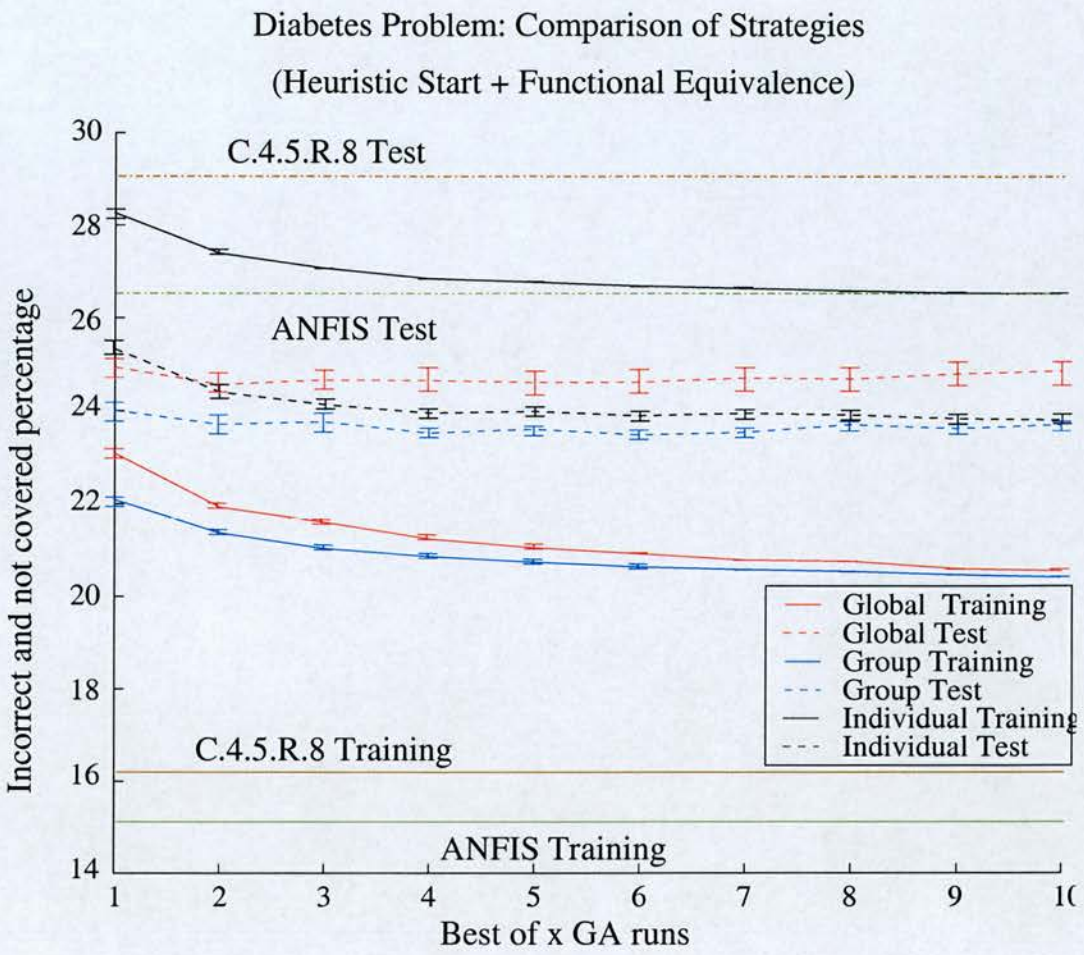


Figure 8.8: Classification error vs translation strategy for the Diabetes problem. Error bars show the 99% confidence interval where applicable.

As group and global strategies allow for more data to be covered, there is more pressure to reduce the number of rules using either of these strategies than using the individual strategy. However, this difference in pressure can be compensated to a certain extent as more general strategies have more evaluations to run. The strong point of the individual strategy rests in the fact that it leads to very short and compact rules as can be seen in figure 8.9.

The pressure on the number of rules objective can be used to reduce the size of the resultant ruleset. This will make it easier to understand the general behavior

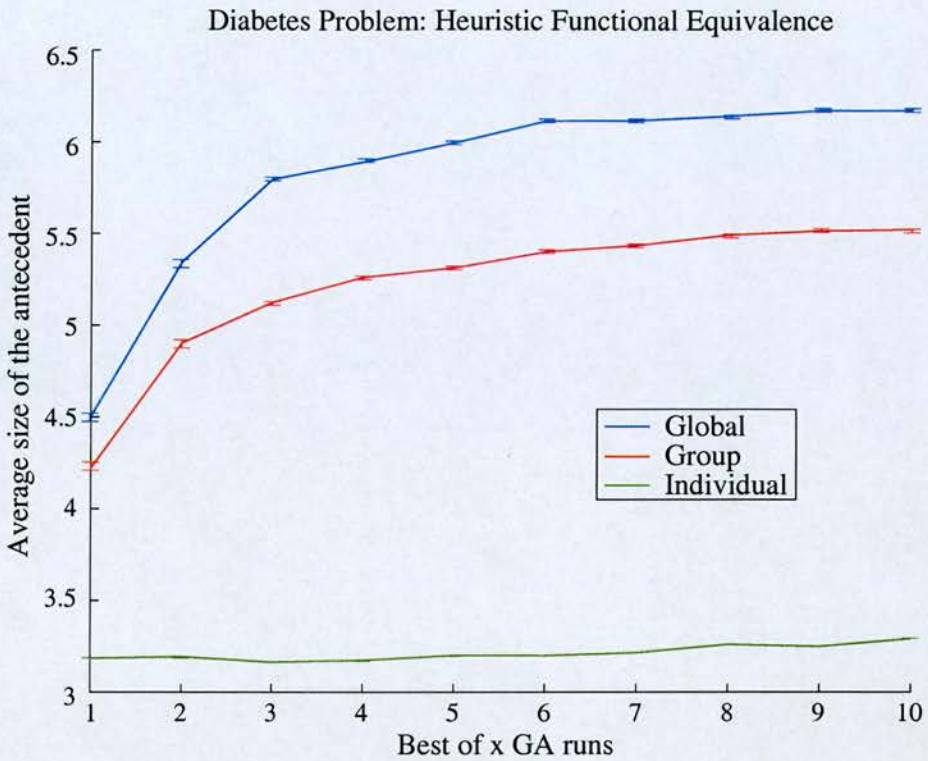


Figure 8.9: Mean size of antecedents vs translation strategy

of the particular classification system under consideration. Nevertheless, in general, a reduced number of rules is likely to result in a reduced classification rate (see figures 8.10 and 8.11 or 8.8 and 8.12).

As different translation strategies lead to very similar testing performances of the resulting descriptive rulesets, only those results obtained by the use of group strategy are hereafter presented.

8.7.2 Effects of heuristic translation

Tables 8.11 and 8.12 give the results with respect to the following (where Train, Test, Rules and Size of Rules respectively stand for the classification error percentage over training data, that over testing data, the number of rules generated and the average number of conditions in the antecedents of the rules):

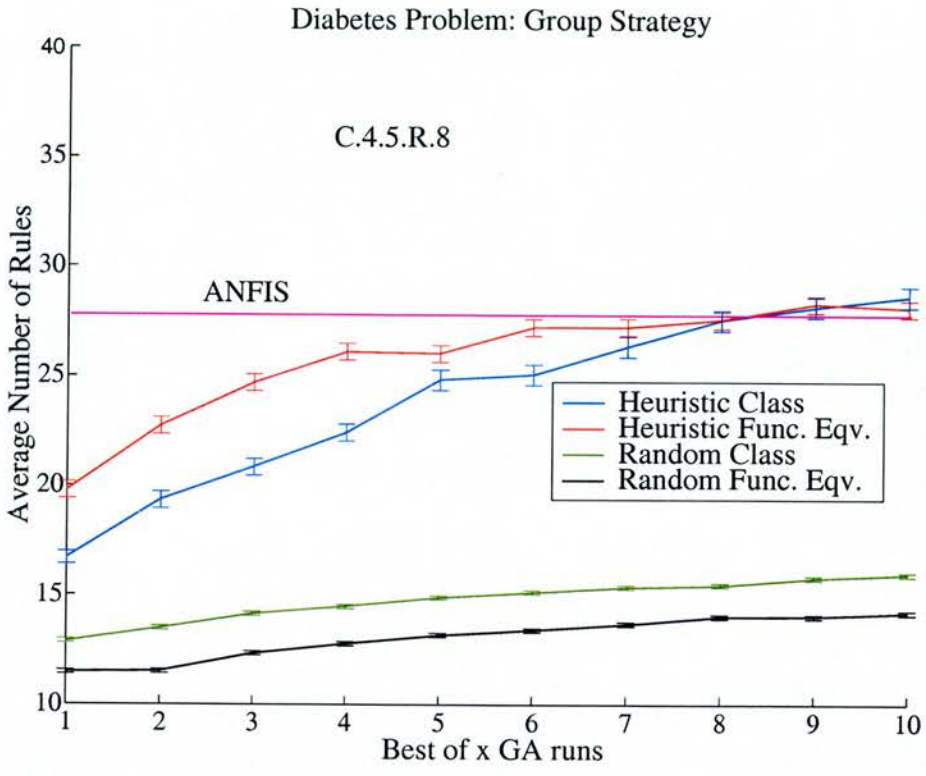


Figure 8.10: Number of rules using the group strategy for the Diabetes problem. Error bars show the 99% confidence interval where applicable.

a) Starting from a heuristic translation and guiding the GA by classification objectives only.

b) Starting from a heuristic translation and guiding the GA by *functional equivalence* and classification objectives.

c) Starting from randomly generated rules and guiding the GA by classification objectives only.

d) Starting from randomly generated rules and guiding the GA by *functional equivalence* and classification objectives.

Within these tables, the most interesting points to compare are those given in the

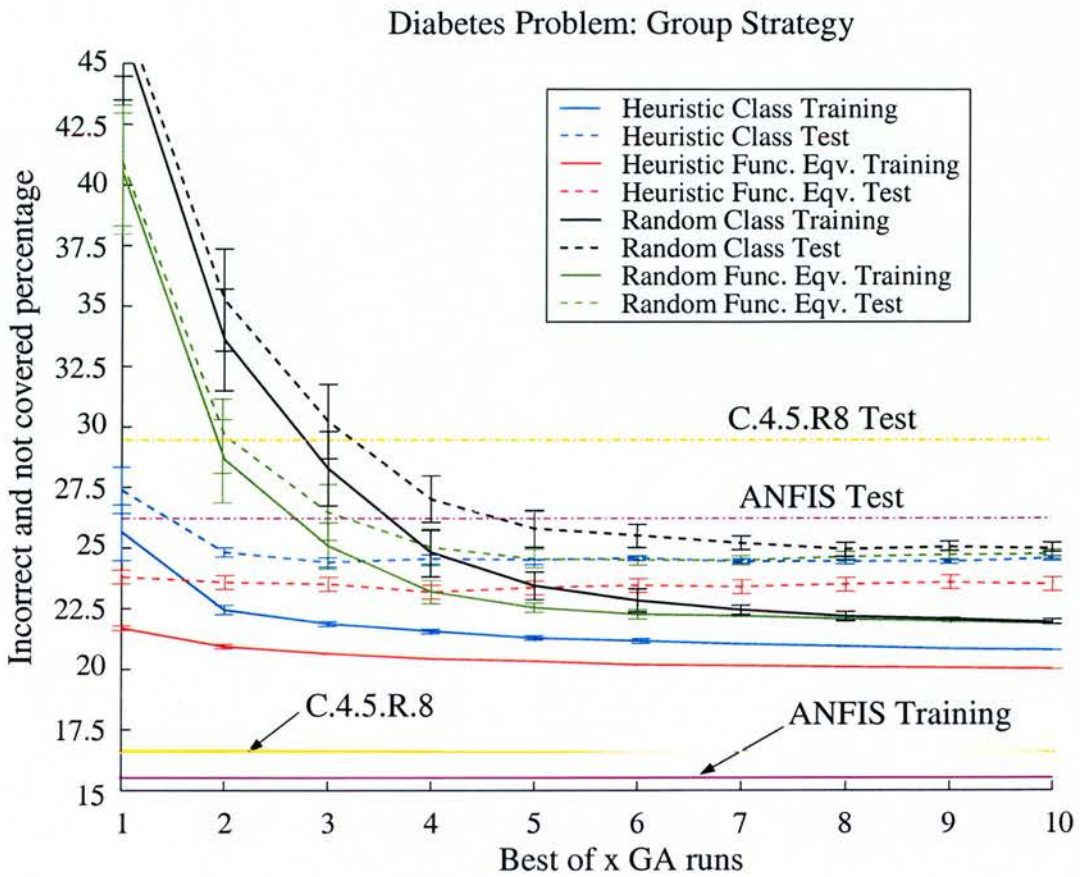


Figure 8.11: Classification error for the Diabetes problem using the group strategy. Error bars show the 99% confidence interval where applicable.

columns concerning items b) and c) above. The former shows the result of what is suggested in this work, and the later shows that of pure data-driven search (no heuristic translation nor approximative model provided). Note that results on the Diabetes problem were collected with two different weights set for the *number of rules* objective, in order to illustrate the impact of this objective upon the translation.

Figure 8.11 shows a graphical comparison, for the Diabetes problem, of the translation results with respect to the number of GA runs. It reveals the difference between different ways of guiding the GA, either starting from a set of descriptive rules obtained by the heuristic translation of the approximative rules or from a set of

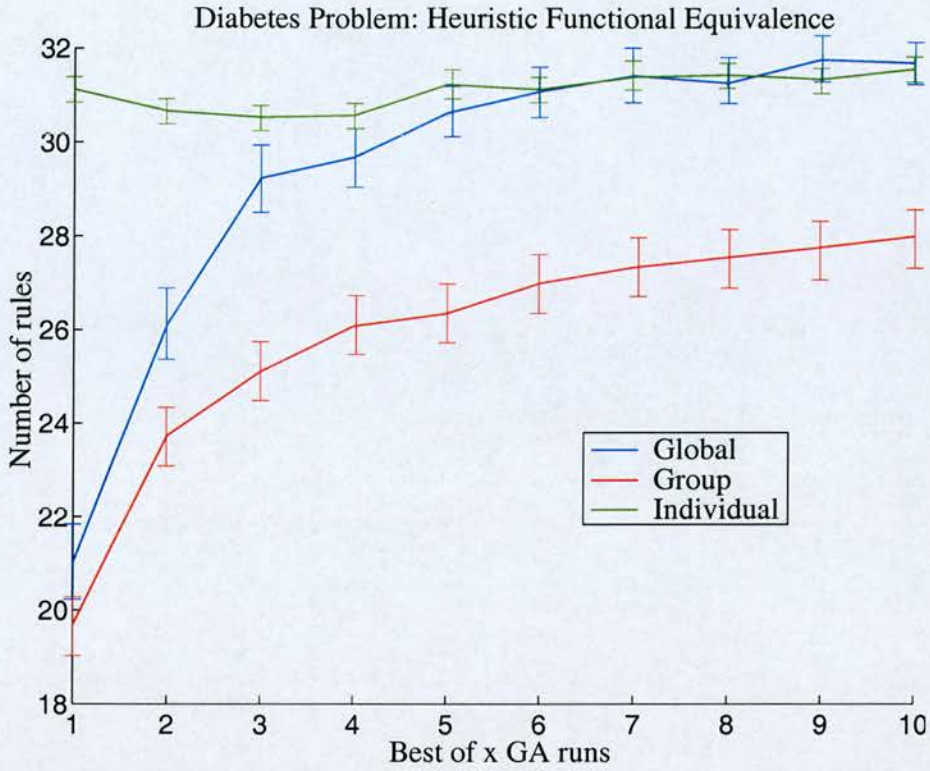


Figure 8.12: Number of rules vs translation strategy for the Diabetes problem. Error bars show the 99% confidence interval where applicable.

randomly generated descriptive rules. Graphs plotting the classification errors for the different classification problems appear to be almost the same in their general tendency. They only differ in the actual values of the classification error and in how many GA runs are needed to reach a state, where running more GAs will not improve the result. These graphs are therefore omitted here.

This general trend shows that a GA-based translation starting from random rules produces systematically worse results than that starting from the heuristic translation. In addition, the results achievable with an initial heuristic translation start are far more stable than those obtained with a random start. This supports the need for generating such first crude translation. For simpler problems (e.g., Breast Cancer) both starting points may eventually lead to similar classification accuracy, yet several runs are

needed to ensure this similarity. Nevertheless, as shown in table 8.11, if only a single run is executed (say, due to computational resource limit) the results obtained using a randomly generated initial population would be much worse than those obtained using the initial population produced by the heuristic method. It clearly pays off to generate the heuristic translation first and to use such a descriptive rule set to act as the generator for the initial population of the GA.

Guidance for the GA by classification error only is also, in general, rather unstable, when compared with the use of *functional equivalence* guidance. This comparison is shown in figures 8.13 and 8.14, where results of 100 runs on classifying the Thyroid problem, with a heuristic start and using the group strategy, are depicted. Following the guidance by classification error alone produces rather poor runs with high error peaks, which actually happens in all tested problems. This unstability of the results can be partially overcome with a higher number of GA runs, though those extra runs are not always affordable computationally. Even if extra GA runs were affordable it would still be better to use the criterion of *functional equivalence* over all the runs.

The above results of a) starting from a heuristic translation and b) making use of *functional equivalence* reveal an important point. That is, generating an approximative model first (to get required *functional equivalence* objectives) and then implementing a heuristic translation (to act as the initial population generator for the GA) improves significantly the final descriptive ruleset.

A positive side effect of the way that the rules are codified in the GA is that the size of the antecedent, i.e. the number of conditions in the antecedent, is variable. This implies that there is an implicit attribute reduction going along with the GA-based optimization as shown in figures 8.9 and 8.15. This reduction is more evident in high dimensional problems. If desired, the reduction of antecedent conditions may even be explicitly introduced as another optimization objective.

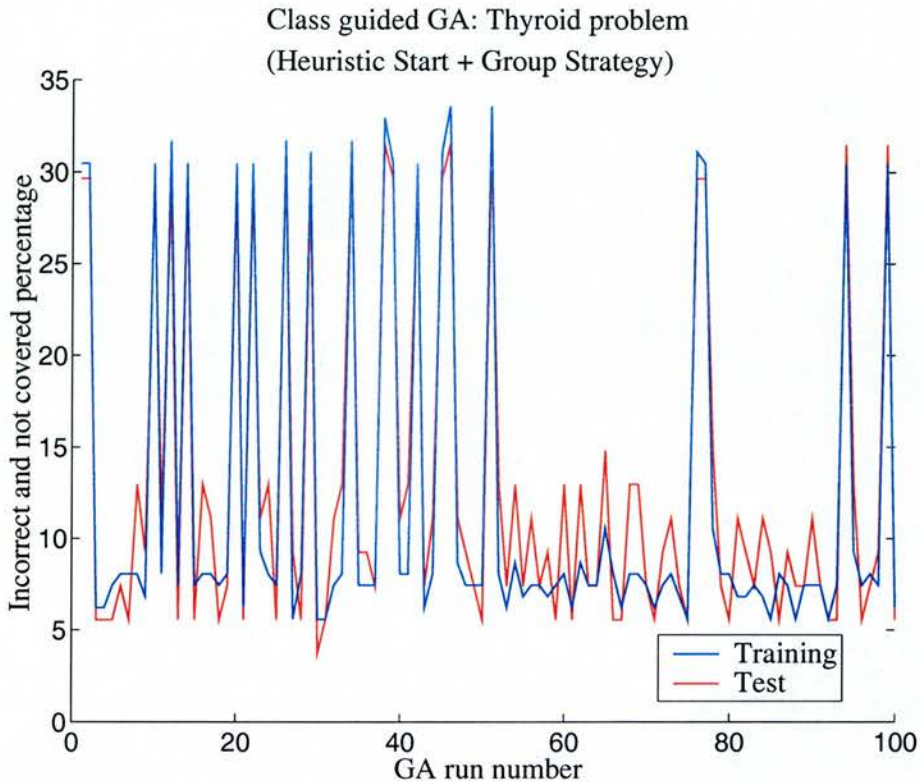


Figure 8.13: 100 runs for class guided guided GA on Thyroid

8.8 Comparison to models produced by alternative approaches

To support comparative studies, results of applying a descriptive ruleset which is obtained by the initial heuristic translation alone, and those of using C4.5, ANFIS and Lozowski's algorithm are also provided as given in tables 8.13 and 8.14. Note that figures 8.8 and 8.10 also show some such experimental results. The work presented in this thesis performs well and does so consistently in testing. The accuracy of translated descriptive models is close to that achievable by the optimized ANFIS (comparing table 8.12 and the ANFIS column of table 8.13), and generally outperforms the other descriptive modeling techniques tested (comparing table 8.12 and table 8.14).

Finally, it is worth noting that, the performance of Lozowski's algorithm, which is

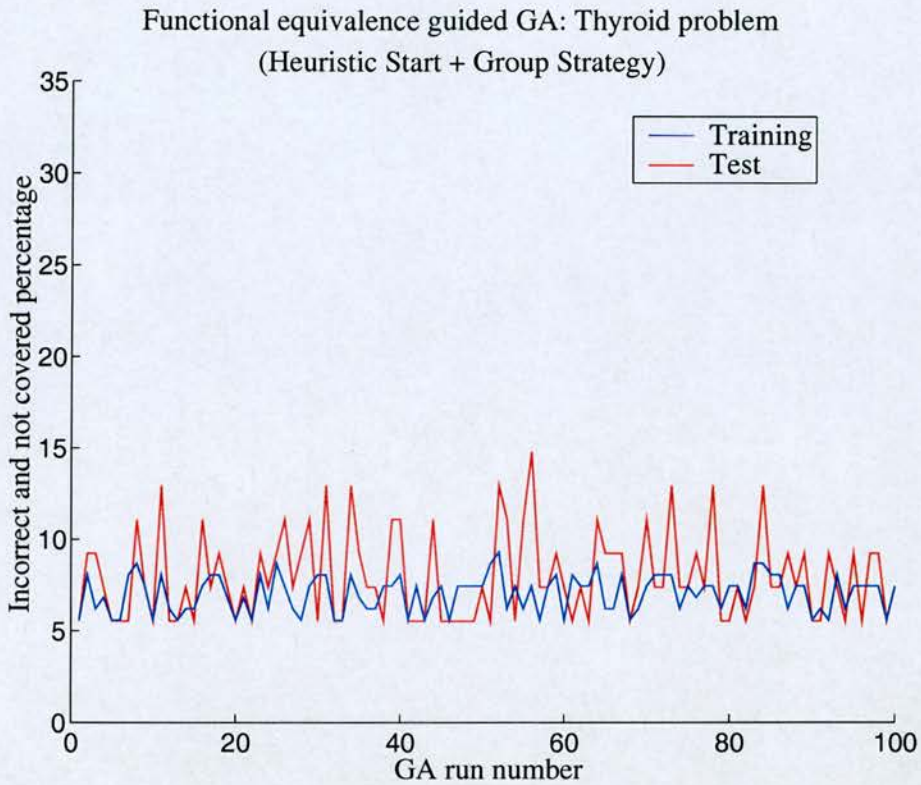


Figure 8.14: 100 runs for functional equivalence guided GA on Thyroid

an exhaustive search based method, never gets close to that of ANFIS or C4.5. In all the problems tested this algorithm gives poorer results than the approximative modeling methods used and yet requires more rules even for reaching such less desirable results. Compared just to the heuristic translation Lozowski's algorithm only defeats it for the Breast Cancer and Iris problems, but its corresponding translated models contain a considerably higher number of rules. However, to be fair with this comparison it must be remembered that Lozowski's algorithm makes use of no hedges.

8.9 Summary

Experimental proof of the effectiveness of the proposed translation mechanism has been provided in this chapter. It started with explaining the various problems

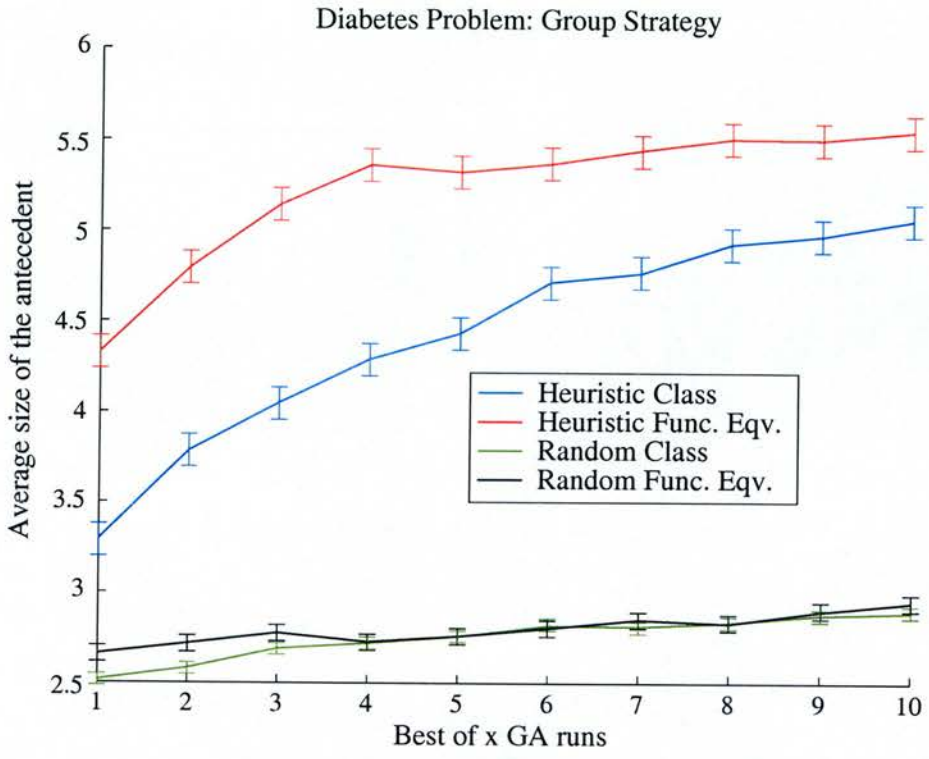


Figure 8.15: Mean size of the antecedents for the Diabetes problem using the group strategy. Error bars show the 99% confidence interval where applicable.

used as benchmark for comparative studies. In particular, the experimental studies demonstrated that the new hedges proposed outperform the classical ones, and that the translated rule models are indeed of high transparency. They also show that modelling accuracy is attained over different problems tested. The results have highlighted the advantages of the proposed approach for generating descriptive fuzzy models as compared to typical existing approaches.

Table 8.11: Mean results, with 99% confidence intervals, of translations using one GA run only (Group Strategy)

Problem	Heuristic / Classification			
	Train Error (%)	Test Error (%)	Rules	Size of Rules
Breast Cancer	2.94 % ± 0.91	7.30 % ± 0.86	9.45 ± 0.15	2.43 ± 0.06
Diabetes (Norm)	25.14 % ± 0.87	26.83 % ± 0.86	16.43 ± 0.59	3.27 ± 0.09
Diabetes (High)	28.74 % ± 1.31	29.72 % ± 1.29	8.28 ± 0.18	2.74 ± 0.07
Iris	1.34 % ± 0.11	4.46 % ± 0.21	5.41 ± 0.09	2.94 ± 0.04
New Thyroid	12.29 % ± 0.78	13.01 % ± 0.74	7.25 ± 0.11	3.13 ± 0.04
Wine	4.75 % ± 1.1	12.52 % ± 1.12	10.21 ± 0.21	3.48 ± 0.11
	Heuristic / Functional Equivalence			
Breast Cancer	1.10 % ± 0.02	5.58 % ± 0.05	8.49 ± 0.15	3.00 ± 0.08
Diabetes (Norm)	21.89 % ± 0.10	24.46 % ± 0.21	19.81 ± 0.79	4.23 ± 0.12
Diabetes (High)	25.39 % ± 1.05	27.24 % ± 0.99	9.10 ± 0.61	3.31 ± 0.09
Iris	1.11 % ± 0.07	4.44 % ± 0.21	5.23 ± 0.1	2.92 ± 0.04
New Thyroid	6.99 % ± 0.08	7.88 % ± 0.19	6.24 ± 0.09	3.6 ± 0.04
Wine	1.51 % ± 0.09	5.55 % ± 0.27	10.8 ± 0.2	4.65 ± 0.11
	Random / Classification			
Breast Cancer	14.93 % ± 2.94	18.30 % ± 2.68	9.71 ± 0.17	2.60 ± 0.06
Diabetes (Norm)	46.61 % ± 2.12	47.29 % ± 2.01	11.55 ± 0.25	2.68 ± 0.06
Diabetes (High)	48.14 % ± 2.19	48.36 % ± 2.03	9.42 ± 0.29	2.62 ± 0.03
Iris	6.05 % ± 1.13	9.79 % ± 1.06	5.15 ± 0.03	2.51 ± 0.03
New Thyroid	53.66 % ± 3.25	54.47 % ± 3.20	7.19 ± 0.11	2.94 ± 0.06
Wine	45.49 % ± 3.09	47.53 % ± 2.82	10.13 ± 0.17	2.32 ± 0.06
	Random / Functional Equivalence			
Breast Cancer	10.04 % ± 2.34	14.02 % ± 2.13	9.18 ± 0.18	2.98 ± 0.08
Diabetes (Norm)	41.53 % ± 2.04	41.91 % ± 1.97	11.52 ± 0.22	2.51 ± 0.05
Diabetes (High)	45.67 % ± 2.10	46.05 % ± 2.09	3.43 ± 0.11	1.57 ± 0.04
Iris	3.28 % ± 0.59	5.90 % ± 0.59	5.17 ± 0.08	2.42 ± 0.03
New Thyroid	45.92 % ± 3.22	46.91 % ± 3.15	6.12 ± 0.07	3.98 ± 0.08
Wine	36.50 % ± 3.10	39.71 % ± 2.80	10.69 ± 0.19	2.68 ± 0.05

Table 8.12: Mean results, with 99% confidence intervals, of the best translation out of 5 GA runs (Group Strategy)

Problem	Heuristic / Classification			
	Train Error (%)	Test Error (%)	Rules	Size of Rules
Breast Cancer	0.76 % ± 0.01	5.75 % ± 0.07	9.33 ± 0.14	2.62 ± 0.06
Diabetes (Norm)	21.34 % ± 0.07	24.49 % ± 0.19	23.81 ± 0.90	4.45 ± 0.11
Diabetes (High)	21.81 % ± 0.08	25.10 % ± 0.17	9.83 ± 0.15	3.10 ± 0.06
Iris	0.51 % ± 0.03	5.29 % ± 0.23	5.90 ± 0.08	3.00 ± 0.03
New Thyroid	6.47 % ± 0.08	7.14 % ± 0.17	7.51 ± 0.13	3.32 ± 0.03
Wine	0.38 % ± 0.03	7.42 % ± 0.28	10.93 ± 0.25	3.69 ± 0.11
	Heuristic / Functional Equivalence			
Breast Cancer	0.75 % ± 0.01	5.61 % ± 0.05	8.30 ± 0.12	3.15 ± 0.08
Diabetes (Norm)	20.60 % ± 0.04	24.11 % ± 0.27	26.41 ± 0.75	5.32 ± 0.10
Diabetes (High)	20.42 % ± 0.07	24.23 % ± 0.16	10.75 ± 0.68	4.44 ± 0.10
Iris	0.48 % ± 0.03	3.41 % ± 0.12	4.71 ± 0.12	3.02 ± 0.03
New Thyroid	5.89 % ± 0.04	6.75 % ± 0.14	6.19 ± 0.09	3.87 ± 0.04
Wine	0.31 % ± 0.04	4.22 % ± 0.32	10.50 ± 0.18	5.20 ± 0.09
	Random / Classification			
Breast Cancer	0.90 % ± 0.24	5.76 % ± 0.23	9.11 ± 0.12	2.65 ± 0.06
Diabetes (Norm)	23.37 % ± 0.51	25.80 % ± 0.53	13.21 ± 0.24	2.81 ± 0.05
Diabetes (High)	23.32 % ± 0.72	26.01 % ± 0.67	8.42 ± 0.25	2.73 ± 0.03
Iris	0.99 % ± 0.05	6.18 % ± 0.12	5.76 ± 0.10	2.68 ± 0.04
New Thyroid	12.03 % ± 1.32	13.88 % ± 1.33	7.30 ± 0.11	3.21 ± 0.06
Wine	6.97 % ± 1.26	13.36 % ± 1.21	10.03 ± 0.16	2.29 ± 0.05
	Random / Functional Equivalence			
Breast Cancer	0.76 % ± 0.01	5.68 % ± 0.07	8.92 ± 0.20	3.12 ± 0.07
Diabetes (Norm)	22.41 % ± 0.15	24.48 % ± 0.20	15.06 ± 0.23	2.81 ± 0.04
Diabetes (High)	23.42 % ± 0.51	24.87 % ± 0.55	4.26 ± 0.12	1.93 ± 0.05
Iris	0.80 % ± 0.04	4.61 % ± 0.17	5.81 ± 0.08	2.42 ± 0.03
New Thyroid	8.36 % ± 0.79	10.57 % ± 0.78	6.27 ± 0.09	4.06 ± 0.07
Wine	2.37 % ± 0.63	10.08 % ± 0.64	10.77 ± 0.21	2.74 ± 0.05

Table 8.13: Classification error and number of rules of some approximative modeling methods

Problem	C4.5			ANFIS		
	Train	Test	Rules	Train	Test	Rules
Breast Cancer	1.2 %	7.6 %	29	0.4 %	4.1 %	18
Diabetes	16.1 %	29.2 %	35	15.7 %	26.6 %	28
Iris	1.8 %	2.6 %	5	0.8 %	2.6 %	4
New Thyroid	1.9 %	7.4 %	13	3.1 %	1.8 %	4
Wine	0.8 %	2.2 %	17	0 %	2.2 %	6

Table 8.14: Classification error and number of rules of Lozowski's method and initial translation

Problem	Lozowski			Extended Heuristic 2		
	Train	Test	Rules	Train	Test	Rules
Breast Cancer	10.3 %	15.2 %	74	25.7 %	23.9 %	94
Diabetes	38.2 %	39.6 %	65	33.5 %	32.8 %	43
Iris	4.5 %	10.5 %	11	8.9 %	11.6 %	9
New Thyroid	82.6 %	83.3 %	4	25.5 %	25.9 %	4
Wine	39.8 %	44.4 %	21	30.1 %	22.2 %	23

Chapter 9

Conclusions

"It is not enough to prove something, one also has to seduce or elevate people to it. That is why the man of knowledge should learns how to speak his wisdom: and often in such a way that it sounds like folly!"

Friedrich Nietzsche

9.1 Summary of the Thesis

This chapter concisely summarizes the most significant contributions of this research and points out important further work. In particular this section reviews the aims of the project and shows what goals and how they were achieved. The next section suggests investigations that can be originated from the results of this thesis.

9.1.1 What was intended to be accomplished

Computing with words is a fundamental contribution of fuzzy logic. For many applications, there exists a considerable volume of historical data obtained by observing the behavior of the system concerned. It is therefore desirable to be able to automatically generate rules from such given data.

Many techniques exist for this, most of which follow the so-called approximative approach, which works by creating and tuning the fuzzy rule bases to best fit the data. The methods obtain accurate models in a fast and efficient way. Unfortunately, they tend to create fuzzy sets that fit the data well but that usually lack features which are considered important to make it easy for human users to interpret the resulting model and its reasoning.

Opposing approximative modeling stands the *descriptive approach* (or linguistic fuzzy modeling). From this point of view, semantics are as important as accuracy. The definition of the fuzzy sets is human given. Unfortunately, current descriptive rule induction algorithms are rather inefficient and slow, mostly based on exhaustive search. Thus, the question became if there would exist a way to generate descriptive rules using a variation of the approximative methods or taking advantage of them.

9.1.2 How was the goal to be achieved

The proposed method to obtain a fast and accurate pure descriptive model from given data was to use a two-step mechanism. The first step is to use an approximative method to create accurate rules and the second to convert the resulting approximative rules to descriptive ones. The conversion is achieved first by employing a heuristic translation to derive potentially useful translations from the approximative fuzzy model, and then by performing a fine tuning of these translations via evolutionary computation, in particular, a genetic algorithm. The resultant descriptive system is ready to be directly applied for inference; no approximative rules are needed in runtime.

To allow more flexible modeling results, such that predefined fuzzy sets would be modified in their effects, within the resulting descriptive rules, without changing their definition, linguistic hedges are used.

9.1.3 What have been achieved

The most important point of this thesis is that the proposed mechanism to obtain a fast and accurate pure descriptive model is successful, as experimentally demonstrated [Marín-Blázquez & Shen 02a]. Not only can substantially better descriptive models be generated in terms of accuracy and transparency, but the method is computationally affordable, unlike existing approaches to descriptive modelling.

Alongside with this main achievement, other contributions made include a new way of calculating hedges than that existed in the literature and the introduction of novel hedges. The classical way of applying the hedges, usually by powering the membership values, was unintuitive and, in the particular case of trapezoidal fuzzy sets, very inefficient, preventing them from practical applications. The new hedges, on the other hand, have proven not only useful but also better capturing the ideas behind the hedges [Marín-Blázquez & Shen 01].

New heuristic methods for translating rules have been created in this thesis also. Suggestions and alternatives are provided in order to deal with the potential explosion of the number of nodes in the rule generation graph used by the heuristic, via similarity-based node reduction mechanisms. Although of limited success by themselves, they could be improved further, but these heuristic methods are adapted to support a more effective translation mechanism to work.

This more effective approach is based on the proposed *functional equivalence* objectives for rule translation/tuning, implemented with a multi-objective GA. As a positive side-effect of applying this approach, ruleset complexity can be reduced by eliminating conditions in the antecedents as, instead of mapping exactly a rule itself, it is its firing pattern that is to be functionally achieved. Thus, rules that correctly classify a point are allowed to fire with even greater strength than the original approximative firing. This makes the GA-based optimization of the emerging descriptive models

easier and more robust. These results have been published in the prestigious IEEE Transactions on Fuzzy Systems [Marín-Blázquez & Shen 02a].

An specialized GA has been designed and implemented to optimize the translation. Special operators designed to address inefficiencies inherent to rule systems were suggested and tested, with very promising results. The work has included a robust GA representation that allows for a degree of freedom in expressing the structure of the rules themselves and the number of rules present in the rule set [Marín-Blázquez & Shen 02c].

The research of this thesis also inspired the consideration of the kind of limits that should be imposed over the modification of the definitions of the membership functions if such a modification would be allowed [Marín-Blázquez & Shen 02b]. A micro-tuning that enforces a very high similarity with the original definition of the fuzzy sets (and hence the concepts they represent) may be supported when desired.

9.2 Future Work

Any work that extends for the span of time and the range of research topics as this thesis usually opens many doors for further investigation than can be traversed or even closed. In the following some such open issues are addressed.

9.2.1 Approximative rule generators

The modeling technique used to create the approximative fuzzy rulesets may create fuzzy models with certain features that make them difficult (or easy) to translate. ANFIS is employed here, but other rule generation techniques may be used for the same purpose. Perhaps, some may be modified to make their reasoning procedure more similar to pure descriptive modelling, thereby creating rules that are, a priori, easier to translate. It would be interesting to test if this conjecture is true.

9.2.2 New hedges applicable to other types of membership functions

One line of future work lies in the generalization, of the new approach to hedges proposed in this thesis. Currently, only trapezoidal fuzzy sets are considered. Yet the concepts of hedges ought to be applicable to other types of fuzzy membership function. General hedge expressions able to transform any possible membership function, including curves, would be very useful. Of course, experimental studies would have to be performed in order to check if the substantial improvement, proven in this thesis of the new hedges over the trapezoidals, could be maintained for different membership functions.

Also, it would be interesting to investigate different combinations of hedge and sets, say, to check if changing a particular hedge for another would produce a highly similar set as modified by the original hedge. If a kind of order and consistency could be detected such information would be of utmost importance to create a move operator designed to make fine grain search. A hill climber with such an operator would be a useful addition, transforming the GA into a memetic GA.

9.2.3 Selective subsets of examples

One of the important issues with supervised learning is that, in most cases, the potential solution must be tested against a large number of cases. In performing classification, for example, to obtain the percentages of correct, incorrect and uncovered elements all testing data points must be classified. The task of classifying all of them each time an evaluation must be done can be too costly to make an effective search. Therefore some kind of data preprocessing technique may be necessary to be applied. This is particularly so, when the number of input variables becomes large. Thus, a dimensionality reduction mechanism to compress the data set will provide further assistance for the present work to be successful in such complex domains.

Note that a a simple but effective method to select only subgroups of data points to perform the evaluation has been reported in [Gathercole & Ross 94]. There, the data points to be chosen to perform an evaluation are selected according to a) how difficult they are to classify, and b) how much time that has passed without them being selected. The first block helps ensure that the search is focused in zones not yet optimized and the second block ensure that what has been learnt is not forgotten. However, the problems tested here are small enough not to have a real need of these techniques. In scale-up applications of the present work, these techniques may became indispensable. this remains as an important further research.

9.2.4 Translating other models

In principle, models created by any technique (even different than a fuzzy model or even rules) that produces firing-strength-like information for a given point or class (so that the functional equivalence objectives may be followed) can be translated using the procedure explained in this thesis. In fact, after the enrichment process described (in section 5.2), the original approximative model is discarded. Although the heuristic translation (that works on the approximative model) plays an important role in providing stability and helpful boost to the GA performance it is not necessary in theory. Therefore, no theoretical obstacle is foreseen to prevent the use of the present work to translate different types of models.

Bibliography

[Abe & Thawonmas 97] Shigeo Abe and Ruck Thawonmas. A fuzzy classifier with ellipsoidal regions. *IEEE Transactions on Fuzzy Systems*, 5(3):358–368, August 1997.

[Abe 98] Shigeo Abe. Dynamic cluster generation for a fuzzy classifier with ellipsoidal regions. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 28(6):869–876, December 1998.

[Abe 99] Shigeo Abe. Fuzzy function approximators with ellipsoidal regions. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics*, 29(4):654–661, August 1999.

[Acan 02] Adnan Acan. GAACO: A GA + ACO hybrid for faster and better search capability. *Lecture Notes in Computer Science*, 2463:300–??, 2002.

[Akbarzadeh-T *et al.* 98] M. R. Akbarzadeh-T, E. Tunstel, K. Kumbla, and M. Jamshidi. Soft computing paradigms for hybrid fuzzy controllers: experiments and applications. In *Proceedings on IEEE International Conference on Fuzzy Systems*, number 2, pages 1200–1205, 1998.

[Antonisse 89] H. J. Antonisse. A new interpretation of schema notation that overturns the binary encoding constraint. In Morgan Kaufmann Publishers, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 86–91, 1989.

[Au & Chan 98] W. Au and K. Chan. An effective algorithm for discovering fuzzy rules in relational databases. In *Proceedings of the 7th IEEE International*

Conference on Fuzzy Systems, pages 1314–1319, 1998.

[Bäck & Schwefel 93] Th. Bäck and H.-P. Schwefel. An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[Barto *et al.* 83] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptative elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13:834–846, September 1983.

[Bentley & Wakefield 96] P. J. Bentley and J. P. Wakefield. An Analysis of Multiobjective optimization within Genetic Algorithms. Technical Report ENGPJB96, University of Huddersfield, UK, 1996.

[Bentley 99] P. J. Bentley. *Evolutionary Design by Computers*. Academic Press Ltd, London, 1999.

[Berenji & Khedkar 92] Hamid R. Berenji and P. Khedkar. Learning and tuning fuzzy controllers through reinforcements. *IEEE Transactions on Neural Networks*, 3(5):724–740, 1992.

[Berenji 96] Hamid R. Berenji. Fuzzy q-learning for generalization of reinforecement learning. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, volume 3, pages 2208–2214, 1996.

[Bezdek & Pal 92] James C. Bezdek and Sankar K. Pal, editors. *Fuzzy Models for Pattern Recognition: Methods that Search for Structures in Data*. IEEE, Pascataway, 1992.

[Bonabeau *et al.* 99] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, 1999.

[Booker *et al.* 89] L. B. Booker, D. E. Goldberg, and J. H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40:235–282, September 1989.

[Breiman *et al.* 84] L. Breiman, J. H. Friedman, R. A. Olsen, and C. J. Stone. *Classification and Regresion Trees.* Wadsworth, Belmont, CA, 1984.

[Burke *et al.* 95] E. K. Burke, J. P. Newall, and R. F. Weare. A memetic algorithm for university exam timetabling. In *1st International Conference on the Practice and Theory of Automated Timetabling (ICPTAT'95, Napier University, Edinburgh, UK, 30th Aug - 1st Sept 1995)*, pages 496–503, 1995.

[Cavicchio 70] D. J. Cavicchio. *Adaptive Search using Simulated Evolution.* Unpublished PhD thesis, The University of Michigan, 1970.
Adaptive evolution in a population of devices (pattern recognition programs) is discussed and criteria for effective reproductive plans are presented.

[Chen *et al.* 98] C. L. Chen, S. H. Hsu, C. T. Hsieh, and W. K. Lin. Generating crisp-type fuzzy models from operating data. In *Proceedings of the 7th IEEE International Conference on Fuzzy Systems*, pages 686–691, 1998.

[Chiu 94] S. L. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2(3), 1994.

[Coello 03] Carlos A. Coello. List of references on evolutionary multiobjective optimization, 2003. Available on web: `http://www.lania.mx/~ccoello/EMOO/EMOObib.html`.

[Cordón & Herrera 99] Oscar Cordón and Francisco Herrera. A two-stage evolutionary process for designing tsk rule-based systems. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics*, 29(6):703–715, December 199.

[Cordón *et al.* 01] Oscar Cordón, Francisco Herrera, Frank Hoffmann, and Luis Magdalena. *Genetic Fuzzy Systems.* World Scientific, 2001.

[Cox 94] Earl Cox. *The Fuzzy Systems Handbook.* AP Professional, Cambridge, MA, 1994.

[Darwin 59] Charles Darwin. *On the Origin of Species.* John Murray, London, 1859.

[Dasgupta & McGregor 92] Dipankar Dasgupta and Douglas R. McGregor. Nonstationary function optimization using the structured genetic algorithm. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2 (Proc. 2nd Int. Conf. on Parallel Problem Solving from Nature, Brussels 1992)*, pages 145–154, Amsterdam, 1992. Elsevier.

[Davis 91] Lawrence Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[De Jong 75] K. A. De Jong. *Analysis of behavior of a class of genetic adaptive systems.* Unpublished PhD thesis, University of Michigan, Dept. Computer and Communication Sciences, 1975. Diss. Abstr. Int 36(10), 5140B.

[Deb & Goldberg 89] Kalyanmoy Deb and David E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J. David Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 42–50, George Mason University, June 1989. Morgan Kaufmann.

[Deb 89] Kalyanmoy Deb. Genetic algorithms in multimodal function optimization. Unpublished M.Sc. thesis, University of Alabama, The Clearinghouse for Genetic Algorithms, Tuscaloosa, AL, 1989. TCGA Report No 88002.

[Deb 91] Kalyanmoy Deb. *Binary and floating-point function optimization using messy genetic algorithms.* Thesis (ph.d.), Dept. of Engineering Mechanics, University of Alabama, 1991.

[Delgado *et al.* 98a] M. Delgado, A. F. Gómez Skarmeta, J. Gómez Marín-Blázquez, and H. Martínez Barberá. Fuzzy

hybrid techniques in modeling. *Lecture Notes in Computer Science*, 1415:180–189, 1998.

[Delgado *et al.* 98b] M. Delgado, A. F. Gómez Skarmeta, and F. Martín. A methodology to model fuzzy systems using fuzzy clustering in a rapid-prototyping approch. *Fuzzy Sets and Systems*, 1998.

[Dick *et al.* 99] S. Dick, A. Kandel, and W.E. Combs. Comment on 'combinatorial rule explosion eliminated by a fuzzy rule configuration' [and reply]. *IEEE Transactions on Fuzzy Systems*, 7(4):475–478, August 1999.

[Dickerson & Kosko 96] Julie A. Dickerson and Bart Kosko. Fuzzy function approximation with ellipsoidal rules. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics*, 26(4):542–550, August 1996.

[Dombi 82] J. Dombi. A general class of fuzzy operators, the De Morgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. *Fuzzy Sets and Systems*, 8:149–163, 1982.

[Dorigo *et al.* 96] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41, 1996.

[Dubois & Prade 80] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, New York, NY, 1980.

[Emami *et al.* 98] M. R. Emami, I. B. Türksen, and A. A. Goldenberg. Development of a systematic methodology of fuzzy logic modeling. *IEEE Transactions on Fuzzy Systems*, 6(3):346–361, August 1998.

[Fang *et al.* 94] Hsiao-Lan Fang, Peter Ross, and Dave Corne. A promising hybrid GA/heuristic approach for open–shop scheduling problems. In A. Cohn, editor, *11th European Conf. on Artificial Intelligence*, pages 590–594, Chichester, 1994. John Wiley & Sons, Ltd.

[Fisher 36] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7, Part II:179–188, 1936.

[Fogarty 96] Terence C. Fogarty, editor. *Inoculation to Initialise Evolutionary Search*, volume 1143 of *Lecture Notes in Computer Science*. Springer, 1996.

[Fogel 92] David B. Fogel. *Evolving Artificial Intelligence.* Unpublished PhD thesis, University of California, San Diego, CA, 1992.

[Fogel *et al.* 66] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley & Sons, New York, 1966.
This work has been criticized as involving overly simple evolution of finite automata, but it was of pioneering significance at the time. For a review see citeLindsay68.

[Fonseca & Fleming 95] Carlos M. Fonseca and Peter J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995.

[Forina] M. Forina. Parvus - an extendible package for data exploration, classification and correlation. Technical report, Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy.

[Frank 79] M. J. Frank. On the simultaneous associativity of $F(x,y)$ and $x + y - F(x,y)$. *Aequationes Mathematicae*, 19:194–226, 1979.

[Fukuda & Shibata 94] Toshio Fukuda and Takanori Shibata. Fuzzy-neuro-GA based intelligent robotics. In J. M. Zurada, R. J. Marks, and C. J. Robinson, editors, *Computational Intelligence: Imitating Life*, pages 352–363, New York, 1994. IEEE Press.

[Fullmer & Miikkulainen 92] B. Fullmer and R. Miikkulainen. Using marker-based genetic encoding of neural networks to evolve finite-state behaviour. In F. J. Varela

and P. Bourgine, editors, *Toward a Practice of Autonomous Systems. Proceedings of the First European Conference on Artificial Life*, Cambridge, MA, USA, 1992. MIT Press.

[Gathercole & Ross 94] C. Gathercole and P. Ross. Dynamic training subset selection for supervised learning in genetic programming. *Lecture Notes in Computer Science*, 866:312–321, 1994.

[George 96] Felicity A. W. George. Hybrid genetic algorithms with immunisation to optimise networks of car dealerships. Edinburgh Parallel Computer Centre, 1996. submitted to Studies in Locational Analysis.

[Goldberg & Richardson 87] D E Goldberg and J Richardson. Genetic algorithms with sharing for multi-modal function optimisation. In *Proc of the 2nd Int. Conf. on Genetic Algorithms and Their Applications*, pages 41–, 1987. GOLDBERG87a.

[Goldberg 89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley", Massachusetts, 1989.

[Goldberg 90] David E. Goldberg. A note on Boltzmann tournament selection for genetic algorithms and population–oriented simulated annealing. *Complex Systems*, 4(4):445–460, 1990.

[Goldberg *et al.* 91] David E. Goldberg, Kalyanmoy Deb, and Bradley Korb. Don't worry, be messy. In Lashon B. Belew, Richard K.; Booker, editor, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 24–30, San Diego, CA, July 1991. Morgan Kaufmann.

[Goldberg *et al.* 92a] David E. Goldberg, Kalyanmoy Deb, and James H. Clark. Accounting for noise in the sizing of populations. In Darrell Whitley, editor, *Foundations of Genetic Algorithms Workshop (FOGA-92)*, Vail, Colorado, July 1992.

[Goldberg *et al.* 92b] David E. Goldberg, Kalyanmoy Deb, and James H. Clark. Genetic algorithms, noise, and the sizing

of populations. *Complex Systems*, 6(4):333–362, August 1992.

[Gómez Skarmeta & Jimenez 97] A. F. Gómez Skarmeta and F. Jimenez. Generating and tuning fuzzy rules using hybrid systems. In *Proceedings of the 6th IEEE International Conference on Fuzzy Systems*, volume 1, pages 247–252, 1997.

[Gómez Skarmeta & Jimenez 99] A. F. Gómez Skarmeta and F. Jimenez. Fuzzy modeling with hybrid systems. *Fuzzy Sets and Systems*, 104(2):199–208, 1999.

[Gómez Skarmeta *et al.* 99] A. F. Gómez Skarmeta, F. Jimenez, and J. Iba nez. Pareto-optimality in scheduling problems. *Lecture Notes in Computer Science*, 1625:177–185, 1999.

[Gómez Skarmeta *et al.* 01] A. F. Gómez Skarmeta, Mercedes Valdés, Fernando Jiménez, and J. Gómez Marín-Blázquez. Approximative fuzzy rules approaches for classification with hybrid-ga. *Information Sciences, 136 (1-4) (2001) pp. 193-214*, 136(1-4):193–214, August 2001.

[Hamacher 75] H. Hamacher. über logische verknupfungen unscharfer aussagen und deren zugehörige bewertungs funktionen. In R. Trappl, G. J. Klir, and L. Ricciardi, editors, *Progress in cybernetics and systems research*, volume III, pages 276–288. Hemisphere, New York, 1975.

[Hart & Ross 98] Emma Hart and Peter Ross. A heuristic combination method for solving job-shop scheduling problems. In Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN V*, pages 845–854, Berlin, 1998. Springer. Lecture Notes in Computer Science 1498.

[Hart *et al.* 98] Emma Hart, Peter Ross, and Jeremy Nelson. Solving a real-world problem using an evolving heuristically driven schedule builder. *Evolutionary Computation*, 6(1):61–80, 1998.

[Hayashi *et al.* 92] Isao Hayashi, Hiroyoshi Nomura, Hisayo Yamasaki, and Noboru Wakami. Construction of fuzzy inference rules by neural network driven fuzzy reasoning with learning functions. *International Journal of Approximate Reasoning*, 6:241–266, February 1992.

[Haykin 94] Simon Haykin. *Neural Networks*. Macmillan College Publishing Company, New York, 1994.

[Hecht-Nielsen 90] R. Hecht-Nielsen. *Neurocomputing*. Addison-Wesley, Readin, MA, 1990.

[Heitkötter & Beasley 97] Jörg Heitkötter and David Beasley. The hitch-hiker's guide to evolutionary computation, 97. Available on web: `http://research.de.uu.net:8080/encore/www/`.

[Ho & Lee 00] Shinn-Ying Ho and Kual-Zhen Lee. A simple and fast GA-SA hybrid image segmentation algorithm. In Darrell Whitley, David Goldberg, Erick Cantu-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 718–725, Las Vegas, Nevada, USA, 10-12 July 2000. Morgan Kaufmann.

[Holland 75] J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, 1975.

[Horn & Nafpliotis 93] Jeffrey Horn and Nicholas Nafpliotis. Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[Horn 93] Jeffrey Horn. Finite Markov Chain Analysis of Genetic Algorithms with Niching. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 110–117, San Mateo, California, 1993. Morgan Kaufmann Publishers.

[Horn *et al.* 94a] Jeffrey Horn, David E. Goldberg, and Kalyanmoy Deb. Implicit Niching in a Learning Classifier System: Nature's Way. *Evolutionary Computation*, 2(1):37–66, 1994. Also IlliGAL Report No 94001, 1994.

[Horn *et al.* 94b] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, 1994. IEEE Service Center.

[Howard & D'Angelo 95] Les M. Howard and Donna J. D'Angelo. The GA–P: A genetic algorithm and genetic programming hybrid. *IEEE Expert*, 10(3):11–15, June 1995.

[Ichihashi *et al.* 96] H. Ichihashi, T. Shirai, K. Nagasaka, and Miyoshi T. Neuro-fuzzy id3: A method of inducing fuzzy decision trees with linear programing for maximizing entropy and an algebraic method for incremental learning. *Fuzzy Sets and Systems*, 81:157–167, 1996.

[Ishibuchi *et al.* 97] H. Ishibuchi, T. Nakashima, and T. Morisawa. Simple fuzzy rule-based classification systems perform well on commonly used real-world data sets. In *NAFIPS '97.*, pages 251–256, 1997.

[Ishibuchi *et al.* 99] H. Ishibuchi, T. Nakashima, and T. Murata. Performance evaluation on fuzzy classifier systems for multidimensional pattern classification problems. *IEEE Transactions on Systems, Man and Cybernetics*, B(29):601–618, October 1999.

[Jang 94] J. Jang. Structure determination in fuzzy modeling: A fuzzy cart approach. In *Proceedings IEEE Conference on Fuzzy Systems*, pages 480–485, 1994.

[Jang *et al.* 93] J.-S. R. Jang, Y. C. Lee, and C.-T. Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 4(1):156–159, January 1993.

[Jang *et al.* 97] J.-S. R. Jang, C.-T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing*. Matlab Curriculum. Prentice Hall, 1997.

[Janikow 98] Cezary Z. Janikow. Fuzzy decision trees: Issues and methods. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics*, 28(1):1–14, February 1998.

[Jelonek *et al.* 95] J. Jelonek, K. Krawiec, and R. Slowinski. Rough set reduction of attributes and their domains for neural networks. *Computational Intelligence*, 11(2):339–347, 1995.

[Jin *et al.* 98] Y. Jin, W. von Seelen, and B. Sendhoff. An approach to rule-based knowledge extraction. In *Proceedings of the 7th IEEE International Conference on Fuzzy Systems*, pages 1188–1193, 1998.

[Jin *et al.* 99] Yaochu Jin, Werner von Seelen, and Bernhard Sendhoff. On generating FC^3 fuzzy rule systems from data using evolution strategies. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics*, 29(6):829–845, December 1999.

[Jones 95] Terry Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. Dissertation, The University of New Mexico, Albuquerque NM, May 1995.

[Joo *et al.* 97] Y. H. Joo, H. S. Hwang, K. B. Kim, and K. B. Woo. Fuzzy system modeling by fuzzy partition and GA hybrid schemes. *Fuzzy Sets and Systems*, 86(3):279–288, 1997.

[Jouffe 98] Lionel Jouffe. Fuzzy inference system learning by reinforcement methods. *IEEE Transactions*

on Systems, Man and Cybernetics, 28(3):338–355, August 1998.

[Kallel & Schoenauer 97] Leila Kallel and Marc Schoenauer. Alternative random initialization in Genetic Algorithms. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 268–275, San Francisco, CA, 1997. Morgan Kaufmann.

[Kallel *et al.* 01] L. Kallel, B. Naudts, and C. R. Reeves. Properties of fitness functions and search landscapes. In Leila Kallel, Bart Naudts, and Alex Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, pages 175–206. Springer, Berlin, 2001.

[Kim 97] C. J. Kim. An algorithmic approach for fuzzy inference. *IEEE Transactions on Fuzzy Systems*, 5(4):585–598, November 1997.

[Kim *et al.* 99] Min-Soeng Kim, Sun-Gi Hong, and Ju-Jang Lee. Self-organizing fuzzy inference system by q-learning. In *Procceedings of the Conference in Fuzzy Systems*, volume 1, pages 372–377, 1999.

[Kitano 90] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4(4):461–476, August 1990.

[Kohonen 89] T. Kohonen. *Self-Organization adn Associative Memory*. Springer-Verlag, Berlin, 1989.

[Kosko 94] B. Kosko. Fuzzy systems as universal approximator. *IEEE Transacions on Computers*, 43:1329– 1333, 1994.

[Kosko 97] B Kosko. *Fuzzy Engineer*. Prentice Hall, 1997.

[Koza 90] J. Koza. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Dept. of Computer Science, Stanford University, June 1990.

[Koza 92] J. Koza. *Genetic Programming: On the Programming of Computers by Natural Selection.* MIT Press, Cambridge Massachusetts, 1992.

[Koza 98] John R. Koza. Genetic programming. In James G. Williams and Allen Kent, editors, *Encyclopedia of Computer Science and Technology*, volume 39, pages 29–43. Marcel-Dekker, 1998.

[Koza *et al.* 99] J. Koza, F.H. Bennet III, W. Mydlowec, M.A. Kleane, J. Yu, and O. Stiffelman. Searching for the imposible using genetic programming. In Morgan Kaufmann, editor, *GECCO-99*, pages 1083–1091, Los Altos, CA, 1999.

[Lee *et al.* 96] Wei-Po Lee, John Hallam, and Henrik Hautop Lund. A hybrid GP/GA approach for co-evolving controllers and robot bodies to achieve fitness-specified tasks. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 20-22 1996. IEEE Press.

[Leich 95] D. D. Leich. *A New Genetic Algorithm for the Evolution of Fuzzy Systems.* Unpublished PhD thesis, Dept. of Egineering Science, University of Oxford, 1995.

[Levine 94] David Levine. *A Parallel Genetic Algorithm for the Set Partitioning Problem.* Unpublished PhD thesis, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, USA, May 1994.

[Liu *et al.* 01] Bin-Da Liu, Chuen-Yau Chen, and Ju-Ying Tsao. Desing of adaptive fuzzy logic controller based on linguistic-hedge concepts and genetic algorithms. *Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 31(1):32–53, February 2001.

[Lomborg 94] Bjørn Lomborg. Game theory versus multiple agents: The iterated prisoner's dilemma. In Cristiano Castefranchi and Eric Werner, editors,

Proceedings of the 4th European Workshop on Modelling Autonomous Agents in a Multi-Agent World : Artificial Social Systems, volume 830 of *LNAI*, pages 69–93, Berlin, July 1994. Springer.

[Lozowski *et al.* 96] A. Lozowski, T. J. Cholewo, and J. M. Zurada. Crisp rule extraction from perceptron network classifiers. In *Proceedings of International Conference on Neural Networks*, volume Plenary, Panel and Special Sessions, pages 94–99, Washington, D.C., 1996.

[Luger & Stubblefield 97] George F. Luger and William A. Stubblefield. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison-Wesley, One Jacob Way, Reading, MA 01867-3999, USA, third edition, 1997.

[Lukasiewicz 70] Jan Lukasiewicz. Philosophical remarks on many-valued systems of propositional logic. In Borkowski, editor, *Selected Works*, Studies in Logic and the Foundations of Mathematics, pages 153–179. North Holland, 1970.

[Mahfoud 91] Samir W. Mahfoud. An analysis of Boltzmann tournament selection. Technical Report IlliGAL Report No 91007, University of Illinois, Department of General Engineering, 117 Transportation Building, 104 South Mathews Avenue, Urbana, IL 61801-2996, 1991.

[Mahfoud 92] S. W. Mahfoud. Crowding and preselection revisited. In Reinhard Männer and Bernard Manderick, editors, *Parallel problem solving from nature 2*, pages 27–36, Amsterdam, 1992. North-Holland.

[Mahfoud 93] Samir W. Mahfoud. Simple analytical models of genetic algorithms for multimodal function optimization. Technical Report IlliGAL Report No 93001, University of Illinois, Department of General Engineering, 117 Transportation Building, 104 South Mathews Avenue, Urbana, IL 61801-2996, 1993.

[Mamdani & Assilian 75] E.H Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7:1–13, 1975.

[Marín-Blázquez & Shen 01] J. Gómez Marín-Blázquez and Q. Shen. Linguistic hedges on trapezoidal fuzzy sets: A revisit. In *Proceedings of the 10th IEEE International Conference on Fuzzy Systems*, volume 1, pages 412–415, December 2001.

[Marín-Blázquez & Shen 02a] J. G. Marín-Blázquez and Qiang Shen. From approximative to descriptive fuzzy classifiers. *IEEE Transactions on Fuzzy Systems*, 10:484–497, August 2002.

[Marín-Blázquez & Shen 02b] J. G. Marín-Blázquez and Qiang Shen. Microtuning of membership functions: Accuracy vs. interpretability. In *Proceedings of the 11th IEEE International Conference on Fuzzy Systems, Fuzz-IEEE 2002*, pages 412–415, Honolulu, USA, May 2002. 2002 World Congress on Computational Intelligence.

[Marín-Blázquez & Shen 02c] J. Gómez Marín-Blázquez and Q. Shen. *Trade-off between Accuracy and Interpretability in Fuzzy Rule-based Modelling*, chapter Regaining comprehensibility of approximative fuzzy models via the use of linguistic hedges. To appear, 2002.

[Marín-Blázquez *et al.* 99] J. Gómez Marín-Blázquez, M. Delgado, A. F. Gómez Skarmeta, and H. Martínez Barberá. A multiagent architecture for fuzzy modeling. *Journal of Intelligent Systems*, 14:305–329, March 1999.

[Marín-Blázquez *et al.* 00] J. Gómez Marín-Blázquez, Q. Shen, and A. F. Gómez Skármeta. From approximative to descriptive models. In *Proceedings of the 9th IEEE International Conference on Fuzzy Systems*, pages 829–834, May 2000.

[Mendel 01] J. M. Mendel. Personal Communucation, Dec 2001.

[Miller *et al.* 60] G. A. Miller, E. Galanter, and K.H. Pribam. *Plans and Structure of Behaviour*. Holt, Rinehart and Winston, New York, 1960.

[Miller *et al.* 89] G. F. Miller, P. M. Todd, and S. U. Hedge. Designing neural networks using genetic algorithms. In D. J. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms, George Mason University, June 4-7, 1989, Arlington, VA*, pages 379–384, San Mateo, CA, 1989. Morgan Kaufmanm.

[Mitchell 97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[Mitra & Pal 96] Sushmita Mitra and Sankar K. Pal. Fuzzy self-organization, inferencing and rule generation. *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, 26(5):608–619, September 1996.

[Mitra *et al.* 97] Sushmita Mitra, Rajat K. De, and Sankar K. Pal. Knowledge-based fuzzy mlp for classification and rule generation. *IEEE Transactions on Neural Networks*, 8(6):1338–1350, November 1997.

[Moriarty & Miikkulainen 93] David Moriarty and Risto Miikkulainen. Evolving complex othello strategies using marker-based genetic encoding of neural networks. Technical Report AI93-206, Department of Computer Sciences, University of Texas, Austin, TX, 1993.

[Moscato 89] P. Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA, 1989.

[Mühlenbein 92] H. Mühlenbein. Parallel Genetic Algorithms in Combinatorial Optimization. In R. Sharda O. Balci and S. Zenios, editors, *Computer Science and Operations Research*, pages 441–456. Pergamon Press, New York, 1992.

[Mühlenbein *et al.* 91] H. Mühlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17(6–7):619–632, September 1991.

[Nauck & Kruse 98] D. Nauck and R. Kruse. NEFCLASS-X– a soft computing tool to build readable fuzzy classifiers. Technical Report 3, BT, July 1998.

[Nauck *et al.* 97] D. Nauck, F. Klawonn, and R. Kruse. *Foundations of Neuro-Fuzzy Systems*. John Wiley & sons, 1997.

[Nomura *et al.* 92] H. Nomura, I. Hayashi, and W. Noboru. A Learning Method of Fuzzy Inference Rules by Descent Method. *Proceedings of the 1st IEEE International Conference on Fuzzy Systems*, pages 203–210, 1992.

[Parmee 99] Ian C. Parmee. A review of evolutionary/adaptive search in engineering design. *Evolutionary Optimization*, 1(1):13–39, 1999.

[Pawlak 91] Zdzisław Pawlak. *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer Academic Publishers, Dordrecht, 1991.

[Peirce 31] C. S. Peirce. *Collected Papers of Charles Sanders Peirce*. Harvard University Press, 1931.

[Poloni 95] Carlo Poloni. Hybrid GA for Multi-Objective Aerodynamic Shape Optimization. In G. Winter, J. Periaux, M. Galan, and P. Cuesta, editors, *Genetic Algorithms in Engineering and Computer Science*, pages 397–416. Wiley & Sons, Chichester, 1995.

[Prechelt 96] L. Prechelt. A quantitative study of experimental evaluations of neural network learning algorithms. *Neural Networks*, 9:457–462, 1996.

[Pretzel 95] Lutz Pretzel. Some notes on neural learning algortihm benchmarking. *Neurocomputing*, 1995.

[Quinlan 86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Quinlan 93] J. R. Quinlan. *C4.5: Programs for machine learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[Radcliffe & Surry 94] N. J. Radcliffe and P. D. Surry. Formal Memetic Algorithms. In T. Fogarty, editor, *Evolutionary Computing: AISB Workshop*, volume 865 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin, 1994.

[Rana *et al.* 96] Soraya Rana, Adele E. Howe, L. Darrell Whitley, and Keith Mathias. Comparing heuristic, evolutionary and local search approaches to scheduling. In B. Drabble, editor, *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 174–181. AAAI Press, 1996.

[Rauma 96] T. Rauma. Knowledge acquisition with fuzzy modeling. In *Proceedings of the 5th IEEE International Conference on Fuzzy Systems*, volume 3, pages 1631–1636, 1996.

[Reeves 96] C. Reeves. Hybrid genetic algorithms for bin-packing and related problems. *Annals of Operations Research*, 63:371–396, 1996.

[Ross & Hallam 96] Peter Ross and John Hallam. Connectionist computing. DAI Teaching Paper 21, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, 1996.

[Ross 98] Peter Ross. *Genetic Algorithms and Genetic Programming.* Department of Artificial Intelligence, The University of Edinburgh, 1998.

[Ross *et al.* 02] Peter Ross, Sonia Schulenburg, Javier Marín-Blázquez, and Emma Hart. Hyper-heuristics: Learning to combine simple heuristics in bin-packing problems. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference (Best Paper Award)*, pages 942–948, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[Roubos & Setnes 01] H. Roubos and M. Setnes. Compact and transparent fuzzy models and classifiers through iterative complexity reduction. *IEEE Transactions on Fuzzy Systems*, 9(4):516–524, August 2001.

[Rumelhart *et al.* 86a] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representation by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. Cambridge, MA: MIT Press, 1986.

[Rumelhart *et al.* 86b] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagation. *Nature*, 323(99):533–536, 1986.

[Russell 23] Bertrand Russell. Vagueness. *Australian Journal of Philosophy*, 1, 1923.

[Samuel 59] A. L. Samuel. Some studies in machine learning using the game of checkers. Technical report, IBM J. Res. Develop., 1959.

[Schaffer & Grefenstette 85] J. David Schaffer and John J. Grefenstette. Multi-objective learning via genetic algorithms. In Aravind Joshi, editor, *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 593–595, Los Angeles, CA, August 1985. Morgan Kaufmann.

[Schaffer 85] J. David Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[Schaffer *et al.* 92] J. David Schaffer, Darrell Whitley, and Larry J. Eshelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In *Proceedings of the Conference on Combinations of Genetic Algorithms and Neural Networks*, pages 1–37, 1992.

[Schwefel 65] H. P. Schwefel. Kybernetische evolution als strategie der experimentelen forschung in der stromungstechnik. Unpublished M.Sc. thesis, Technical University of Berlin, March 1965.

[Schwefel 81] H. P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley and Sons, New York, New York, 1981.

[Schweizer & Sklar 63] B. Schweizer and A. Sklar. Associative functions and abstract semigroups. *Publ. Math. Debrecen*, 10:69–81, 1963.

[Setnes & Roubos 00] M. Setnes and H. Roubos. Ga-fuzzy modeling and classification: Complexity and performance. *IEEE Transactions of Fuzzy Systems*, 8(5):509–522, October 2000.

[Setnes *et al.* 98a] M. Setnes, R. Babuska, and H. B. Verbruggen. Rule-based modeling: Precision and transparency. *IEEE Transactions on Systems, Man and Cybernetics - Part c: Applications and Reviews*, 28(1):165–169, Feb. 1998.

[Setnes *et al.* 98b] M. Setnes, R. Babuska, and H. B. Verbruggen. Transparent fuzzy modelling. *International Journal of Human-Computer Studies*, 49(2):159–179, 1998.

[Shen & Chouchoulas 99] Q. Shen and A. Chouchoulas. Combining rough sets and data-driven fuzzy learning for generation of classification rules. *Pattern Recognition*, 32(12):2073–2076, October 1999.

[Shen & Chouchoulas 00] Q. Shen and A. Chouchoulas. A modular approach to generating fuzzy rules with reduced attributes for the monitoring of complex systems. *Engineering Applications of Artificial Intelligence*, 13:263–278, 2000.

[Shen & Leitch 93] Q. Shen and R. Leitch. Fuzzy qualitative simulation. *IEEE Transactions on Systems, Man and Cybernetics*, 23(4):1038–1061, 1993.

[Simpson 92] Patrick K. Simpson. Fuzzy min-max neural networks-part 1: Classification. *IEEE Transactions*

on Neural Networks, 3(5):776–786, September 1992.

[Smith 80] S. F. Smith. *A learning system based on genetic algorithms*. Unpublished PhD thesis, University of Pittsburgh, 1980.

[Srinivas & Deb 94] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.

[Srinivas & Deb 95] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1995.

[Storn & Price 97] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Global OptimizGómez Skarmeta, A. F. and F. Jimenezation*, 11:314–359, 1997.

[Sugeno & Kang 88] M. Sugeno and G. T. Kang. Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 28(1):15–33, 1988.

[Sugeno & Takagi 83] Michio Sugeno and Tomohiro Takagi. Multi-dimensional fuzzy reasoning. *Fuzzy Sets and Systems*, 9(3):313–325, 1983.

[Sugeno & Yasukawa 93] M. Sugeno and T. Yasukawa. A fuzzy logic based approach to qualitative modeling. *IEEE Transactions on Fuzzy Systems*, 1(1):7–31, August 1993.

[Sugeno 77] M. Sugeno. Fuzzy measures and fuzzy integrals: A survey. In Madan M. Gupta, George N. Saridis, and Brian R. Gaines, editors, *Fuzzy Automata and Decision Processes*, pages 89–102, New York, 1977. North–Holland.

[Takagi & Sugeno 85] Y. Takagi and M. Sugeno. Fuzzy Identification of Systems and Its Applications to Modeling and

Control. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-15:116–132, 1985.

[Tanaka 96] Kazuo Tanaka. *An Introduction to Fuzzy Logic for Practical Applications*. Springer, 1996.

[Tsang *et al.* 00] E. C. C. Tsang, X. Z. Wang, and D. S. Yeung. Improving learning accuracy of fuzzy decision trees by hybrid neural networks. *IEEE Transactions on Fuzzy Systems*, 8(5):601–614, October 2000.

[Tunstel *et al.* 96] E. Tunstel, M. R. Akbarzadeh-T, K. Kumbla, and M. Jamshidi. Soft computing paradigms for learning fuzzy controllers with applications to robotics. In *Proceedings on Biennial Conference of the North American Fuzzy Information Procesing Society, NAFIPS*, pages 255–359, 1996.

[Tuson 98] Andrew Tuson. Optimisation with hillclimbing on steroids: an overview of neighbourhood search techniques. Technical Report 883, Department of Artificial Intelligence, The University of Edinburgh, January 1998.

[UCI] Uci machine learning databases. Available on web: `http://ftp.ics.uci.edu/pub/machine-learning-databases/`.

[Umanol *et al.* 94] M. Umanol, H. Okamoto, I. Hatono, H. Tamur, F. Kawachi, S. Umedzu, and J. Kinoshita. Fuzzy decision trees by fuzzy id3 algorithm and its application to diagnosis systems. In *IEEE World Congress on Computational Intelligence. Proceedings of the Third IEEE Conference on*, volume 3 of *Fuzzy Systems*, pages 2113–2118, 1994.

[Valente deOliveira 99] J. Valente de Oliveira. Semantic constrains for membership function optimization. *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans*, 29(1):128–138, Jan. 1999.

[Velasco 98] Juan R. Velasco. Genetic-based on-line learning for fuzzy process control. *International Journal on Intelligent Systems*, 13:891–903, 1998.

[Wang & Hong 98] X. Z. Wang and J. R. Hong. On the handling of fuzziness for continuous-valued attributes in fuzzy decision trees. *Fuzzy Sets and Systems*, 99(3):283–290, 1998.

[Wang & Mendel 92] L. X. Wang and J. M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1414–1427, November-December 1992.

[Wang 92] L.-X. Wang. Fuzzy systems as universal approximators. In *Proceedings of the 1st IEEE International Conference on Fuzzy Systems*, pages 1153–1162, San Diego, CA, March 1992.

[Wang *et al.* 00] X. Z. Wang, B. Chen, Qian G. L., and F. Ye. On the optimization of fuzzy decision trees. *Fuzzy Sets and Systems*, 112:117–125, 2000.

[Wang *et al.* 01] X. Z. Wang, D. S. Yeung, and E. C. C. Tsang. A comparative study on heuristic algorithms for generating fuzzy decision trees. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics*, 31(2):215–226, April 2001.

[Watkins 89] J. C. H. Watkins, C. *Learning from delayed rewards*. Unpublished PhD thesis, University of Cambridge, Cambridge, UK, 1989.

[Whitley & Starkweather 90] D. Whitley and T. Starkweather. GENITOR II: A distributed genetic algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(3):189–214, July-September 1990.

[Whitley 89] Darrell Whitley. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufman.

[Whitley 01] Darrell Whitley. An overview of evolutionary algorithms: practical issues and common pitfalls. *Information and Software Technology*, 43(14):817–831, 2001.

[Wolberg & Mangasarian 90] W. H. Wolberg and O. L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In *Proceedings of the National Academy of Sciences*, volume 87, pages 9193–9196, U.S.A., December 1990.

[Yager & Filev 94] R. R. Yager and D. P. Filev. *Essentials of Fuzzy Modeling and Control*. Wiley, 1994.

[Yager 79] Ronald Yager. On the measure of fuzziness and negation, part I: membership in the unit interval. *International Journal of Man-Machine Studies*, 5:221–229, 1979.

[Yager 80] Ronald R. Yager. On a general class of fuzzy connectives. *Fuzzy Sets and Systems*, 4(3):235–242, 1980.

[Yeung & Tsang 97] D. S. Yeung and C. C. Tsang. A comparative study on similarity-based fuzzy reasoning methods. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 27(2):216–227, April 1997.

[Yuan & Shaw 95] Y. Yuan and M. J. Shaw. Induction of fuzzy decision trees. *Fuzzy Sets and Systems*, 69:125–139, 1995.

[Zadeh 65] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[Zadeh 73] Lotfi A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions of Systems, Man and Cybernetics*, SMC-3:28–44, 1973.

[Zadeh 75] Lofti A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning i. *Information Sciences*, 8:199–249, 1975.

[Zadeh 87] Lotfi A. Zadeh. Similarity relations and fuzzy orderings. In R. R. Yager, S. Ovchinnikov, R. M. Tong, and H. T. Nguyen, editors, *Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh*, pages 81–104, New York, 1987. John Wiley & Sons, Inc.

[Zadeh 96] Lofti A. Zadeh. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 4(2):103–111, May 1996.

[Zwick *et al.* 87] R. Zwick, E. Carlstein, and D. V. Budescu. Measures of similarity among fuzzy concepts: A comparative analysis. *International Journal of Approximate Reasoning*, 1:221–242, 1987.