Data-Driven System Identification
via
Evolutionary Retrieval of Takagi-Sugeno Fuzzy Models

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

vorgelegt der Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von: Dipl.-Inf. Ingo Renners
geb. am 16. August 1969 in Opladen

Soest, den 9. März 2004

Zusammenfassung

Systemidentifikation hat die Aufgabe, eine Anzahl von zusammengehörenden Komponenten der realen Welt in einem Modell abzubilden. Wenn diese Abbildung durch den Transfer von menschlichem Expertenwissen in ein Modell geschieht, wird dies als wissensbasierte Modellierung bezeichnet. Wenn die Informationen über das System allerdings nur implizit und formlos in Datenbeständen vorliegen, wird die Abbildung dieses Wissens mit Hilfe von Algorithmen als datengetriebene Modellierung bezeichnet.

In dieser Arbeit wird vorgeschlagen, für die datengetriebene Systemidentifizierung die Klasse der sogenannten Takai-Sugeno Fuzzy Modelle zu benutzen. Dies wird durch das Vorhandensein effektiver Lernalgorithmen für diese Klasse von Modellen begründet. Des weiteren ist es oft vorteilhaft, die bei der Systemidentifizierung gefundenen Modelle auch interpretieren zu können. Daher wird auf die Formulierung verschiedener Interpretierbarkeitsfaktoren, welche zu einem objektiven und leicht zu implementierenden Interpretierbarkeitsmaß für Takagi-Sugeno-Modelle zusammengeführt werden können, besonderer Wert gelegt.

Um optimale Strukturen der Modelle zu identifizieren, werden neue Konzepte aus dem Bereich der Heuristik, speziell der evolutionären Berechnungsmethoden, als generell nutzbare Suchmethode angewendet. Optimale und schlanke Modellstrukturen sind in Hinsicht auf Genauigkeit, aber insbesondere im Hinblick auf die Generalisierungfähigkeit von Modellen sehr wünschenswert. Allerdings spielt die notwendige Kodierung von potentiellen Modellen innerhalb einer künstlichen Evolution eine bedeutende, wenn nicht sogar die entscheidende Rolle. Aus diesem Grunde wird in dieser Arbeit eine in diesem Zusammenhang neuartige Methode der Kodierung vorgeschlagen. Dabei wird der Suchraum eines evolutionären Algorithmus durch sogenannte Genotyp-Schablonen aufgespannt, welche mit Hilfe einer kontextfreien Grammatik formuliert werden.

Die vorgeschlagene Methode zur Systemidentifizierung mittels Takagi-Sugeno-Modellen wird dann an einem künstlichen und einem komplexen realen Problem getestet. In der realen Problemstellung geht es um die Identifikation von Modellen, welche die Toxizität von Molekülen vorhersagen. Diese Modelle sollen also einen Zusammenhang von einfach zu messenden oder zu berechnenden Eigenschaften von Molekülen, sogenannten molekularen Deskriptoren, zu deren Giftigkeit aufdecken und herstellen.

Abstract

System identification is the task to map several related components of a real world system into a model. If this is done by transferring human expertise into a model, the process is called knowledge-driven modeling. If the system information is embedded in data-bases and the implicit existent expertise is mapped by algorithms into a model, the process is called data-driven modeling.

This thesis suggests for data-driven system identification the class of Takagi-Sugeno fuzzy models as target. This class of models provides the possibility to make use of powerful learning algorithms. On the other hand the human interpretability of the resulting models can be assured.

Because of this, necessary interpretability factors are worked out and an objective interpretability measure for Takagi-Sugeno fuzzy models is formulated.

Evolutionary computation, as a general search method, is used to identify an optimal model structure. Optimal and sparse model structures are desirable for reasons of accuracy and generalization capability. The way in which candidate solutions (i.e. models), are encoded in evolutionary algorithms is a central factor in population based search methods. The author proposes a novel grammar based method to formulate genotype-templates. These templates will be used to define the genotype search space.

The presented approach of data-driven system identification via evolutionary retrieval of Takagi-Sugeno fuzzy models is tested with artificial data and with a complex real world dataset considering the prediction of molecular toxicity.

Acknowledgments

First of all I want to thank my wife. The possibility to get the impression that my real love is a computer was always existent but never true nor accused.

I also want to thank Prof. Grauel, one of the most reliable men I know, for steady support, motivating discussions and holding off much of the bureaucracy many people have to fight with. I consider the granted scientific freedom during my research as a valuable present.

This scientific freedom also was supported by the Ministry of Sciences and Research, North Rhine-Westphalia, through financial support and especially within the European Union project COMET. Concerning this project my special thanks go to Dr. Benfenati who provided me with the newest toxicity dataset used in the successor project IMAGETOX.

I also want to thank Prof. Kruse who recommended this doctorate. Furthermore he and his colleague Dr. Borgelt provided some very helpful suggestions which accounts for the completion of this thesis.

Finally I want to honor the idea of open source software, allowing me and millions of other people to use thousands of algorithms and applications for free.

Contents

List of Figures

List of Tables

List of Abbreviations

AIC	Akaike's information criterion
ASMOD	adaptive spline modeling
ATSFM	approximate Takagi-Sugeno fuzzy model
BIC	Bayesian information criterion
BNF	Backus-Naur form
COMET	computerized molecular evaluation of toxicity
CP	crossover point
DNA	deoxyribose nucleic acid
DTSFM	descriptive Takagi-Sugeno fuzzy model
EA	evolutionary algorithm
EC	evolutionary computation
ES	evolutionary strategies
FM	fuzzy model
GA	genetic algorithm
GIMP	GNU image manipulation program
GLib	G Library
GNU	a recursive acronym for GNU's not Unix
GTK+	GIMP Toolkit
IC	information criteria
IF	interpretability factor
IM	interpretability measure
IMAGETOX	intelligent modelling algorithms for general evaluation of toxicities
LC50	lethal concentration for 50% of a set of individuals of a population after a given amount of time (i.e. 96 hours)
LOLIMOT	local linear model tree

LS	least squares
MCCV	Monte Carlo cross-validation
MF	membership function
mRNA	messenger ribonucleic acid
MSE	mean square error
NP	non-deterministic polynomial-time
QSAR	quantative structure activity relationships
TSFM	Takagi-Sugeno fuzzy model

Chapter 1

Introduction

1.1 Problem Statement and Motivation

In general, *models* are simplified mappings of parts of reality to any kind of material. Today's world is full of models. Each formula is a model, toy cars are models of their big counterparts, the brain together with the senses forms a model of our environment, and thus models specify our behavior. In science and industry models become more and more important, because they are used to understand, control, optimize, predict or simulate real world processes. Beside the intellectual benefits there is obviously a huge potential of capital gains.

The traditional method of modeling is to utilize human expert knowledge and intuition in combination with data obtained by observations, polls or measurements. These data were mainly stored in books and thus they are not directly accessible by computerized processing methods. However, as a consequence of the technical development in computer science, the capacity of electronically based storing and processing data doubles approximately every 18 month. This trend started in the middle of the twentieth century and is unbroken till now. Furthermore, the worldwide cross-linking of computers via the Internet, which started in the 1990s, enabled the possibility to share and process data worldwide. Associated with these developments, the amount of available data reached a level that could not be handled completely by man. Furthermore, it can be assumed that many datasets contain sufficient correlated data to establish new and efficient models of real world processes.

A natural consequence is to try to take advantage of these neglected data with the help of computational processing. This processing is referred to as *data mining* or *knowledge discovery*. Nowadays, most of these approaches focus on handpicked data-sets, which are classified as most probable to bring in invested resources. Logically and wisely all available human expert knowledge should be incorporated into the emerging model. This is comparable to the first stage of a gold rush, which will continue until most of the obviously fruitful claims are exploited. In a second stage the focus will change to the bulk of medium to low profit modeling. Human preprocessing and incorporation of human expert knowledge becomes undesired because of costs. Fully data-driven methods are needed to meet these demands. Nevertheless, the possibility to incorporate human expert knowledge into a data-driven constructed model should be preserved, since a medium profit targeted datamining process could always identify a high profitable model with applications worth to invest human resources.

Having these assumptions in mind, the problem tackled in this thesis is to develop a fully data-driven method for modeling, in order to meet the upcoming requirements of information handling. The title of this thesis was chosen as it is, because a model is, by definition, always related to a real world system and the model template was selected for various reasons as a Takagi-Sugeno fuzzy model.

1.2 Thesis Contributions

The main contribution of this thesis is the development of an extendable framework for automatic and data-driven modeling. Extendable in the sense, that a model class is used that is also comprehensible for humans and not only executable for machines. An important point is that the resulting models can be refined or analyzed by human experts.

To achieve this goal, an objective interpretability measure for the chosen class of Takagi-Sugeno fuzzy models is provided. To keep the framework as general as possible, evolutionary computation was deployed. To stick to the required principles of generality a novel concept of grammar based problem encoding is introduced, presented and applied to artificial and real world data.

1.3 Thesis Structure

This section briefly summarizes the organization of the content and the notation of this thesis. Following the table of content are a list of figures, a list of tables and a list of used abbreviations. The appendix is concerned with the presented real world example[1]. The bibliography can be found at the end of the thesis, followed by an index. Furthermore, it should be mentioned that the availability of the down-loadable versions[2] of the bibliography entries were finally checked in March 2004. Due to the rapid development of the Internet it is possible that some references are no longer reachable.

1.3.1 Organization of the Content

This thesis covers basic concepts and applications of system identification, fuzzy logic and evolutionary computation and their integration synergism. The used concepts are reflected in chapters 2 to 4, always with a focus on the implementation of a data-driven system identification algorithm. In the second half of chapter 4 a novel concept of defining genotype search spaces is presented. This method will be used in the subsequent chapters 5 to 6, which deal with the implementation and testing of the developed system identification algorithm.

Chapter 1 (this chapter) provides a brief summary of the organization and notation used in this thesis.

Chapter 2 contains all used definitions and findings regarding system identification. The chapter start with explanations of the terms system and model. This is followed by a summary of model types and model application areas. The main focus is on the tasks which come along with system identification based on data-driven concepts. Especially mathematical methods for parameter estimation of models are recapitulated, with respect to purely data-driven modeling. Furthermore, the importance and the relation between model complexity and model validation is stressed and several model validation approaches are discussed.

[1] For readers of the electronic pdf version it is worth noting that the graphically presented results of Sec. 6.2 are linked to the appendix.

[2] Only some free available papers were linked with the bibliography. Readers of the electronic version can click on the concerning link to read the referenced paper.

Chapter 3 starts with a basic introduction to fuzzy concepts, different fuzzy models and fuzzy operations. The main focus is on the functionality of Takagi-Sugeno fuzzy models, which will be discussed in greater detail. This covers the selection criteria responsible for choosing Takagi-Sugeno fuzzy models for system identification, methods of parameter and structure learning of Takagi-Sugeno fuzzy models and a detailed discussion of interpretability considerations.

Chapter 4 presents the terminology, possibilities and restrictions of evolutionary computation. The representation (genotype) of candidate solutions is used as the initial point to introduce the main concepts of evolutionary computation. An important part of this chapter proposes and introduces a novel grammar based representation scheme, which provides an often applicable simplification of the in general hard to solve problem encoding task.

Chapter 5 utilizes the concepts which were introduced in the previous chapters, to establish a general framework for data-driven system identification via evolutionary optimized Takagi-Sugeno Fuzzy Systems.

Chapter 6 present the results obtained by applying the in chapter 4 developed and in chapter 5 implemented approach of grammar based solution encoding to an artificial and to a complex real world dataset. In all cases the data were used with different levels of cross-validation to validate and compare the results.

Chapter 7 concludes with a brief summary of achieved results and newly introduced concepts. The extension capabilities of the presented framework is outlined and finally an outlook of interesting future work in the field of data-driven system identification is given.

1.3.2 Notation

As mentioned above, this thesis contains an index with all relevant technical terms stating the page of appearance in the thesis. Terms which are inserted into the index can be identified by their *italic* appearance. Furthermore indexed terms may be set in **bold font** if the term occur in a description or in a (sub)section title. Descriptive names of functions, like `ModelOut`$(\cdot)$, are set in typewriter font, vectors are marked by an underline (e.g. $\underline{w}$) and matrices are written as bold set capital letters (e.g. $\mathbf{X}$). The variable y always denotes the desired model output and $\hat{y}$ the calculated model output. Quotation marks are used if a term (for example ”good”) is not clearly defined.

1.3.3 Summary

This and all subsequent chapters will end with a short outline of the concerning chapter-content.

This chapter provided the problem statement, the motivation to tackle this problem and a short sketch of the main thesis contribution, namely a novel grammar based concept of problem encoding. Furthermore, the organization of the thesis content was given and the applied notation was mentioned.

Chapter 2

System Identification

The term *system* has its origin in the Greek language and can be explained in such a way, that a system consists of several components, which somehow form a whole. The general behavior of a system can be described by some important characteristics, all referring to the *state of a system*. The state of a system describes the system at a certain point in time. Systems with a finite or countable number of system states are called *discrete systems* and systems with an uncountable number of system states are called *continuous systems*. Closely related to these terms are the following system characteristics:

Static systems does not change their system states in time.

Stationary systems are characterized by the fact that their system state changes are constant in time.

Dynamic systems are characterized by the fact that their actual system states are defined by their initial states and the time depending system inputs.

In fact all systems are dynamic (continuous), but many systems can be considered as static by observing a certain time segment. Furthermore, lets define external influences as system state changing factors which are not generated by system parts. By doing so another mutually exclusive characteristic of a system is described by the terms:

Open systems are characterized by the fact that their system states are subject to external influences.

Closed systems are characterized by the fact that their system states are not subject to external influences.

Again, all systems are open[1], but many systems can be considered as closed, because the influence of external factors to the system state is negligible. Each model of a system should be designed in such a way that the external factors affecting the system state of the model are minimized.

System identification is the task to map several related components of the real world into a model. What the term model subsumes and what system identification is used for will be pointed out in the next sections. Because this thesis focuses on data-driven mathematical system identification, Sec. 2.3 provides a brief overview of tasks which has to be performed before starting such a kind of system identification and Sec. 2.4 deals with the necessary parameter optimization of candidate models. This chapter will close with two sections considering generally valid statements about model complexity (Sec. 2.5) and the resulting needs for model validation (Sec. 2.6).

2.1 Model Types

A *model* always imitates the behavior of a real world system. A somewhat rough classification leads to the following four different model types:

- Scaled models
- Flowcharts
- Tables
- Mathematical models

2.1.1 Scaled Models

Scaled models are often used to validate theoretical assumptions. For example a bench-scaled model of a production facility is used to validate if the actual production process is feasible or, for example, a bench-scaled model of a bridge, using new materials, has to undergo severe tests to validate the expected carrying capacity. Scaled

[1]Except the one system subsuming all existing parts (possibly the universe).

models are also used in iterative simulation-optimization processes. For example, the impact of a car-design to the potential clientele and to service ability is tested with a one-to-one model. The response is used to optimize the product.

2.1.2 Flowcharts

A flowchart or a characteristic diagram is used to illustrate the steps in a process. Each box in a flowchart represents a step and each arrow represents the sequence of steps. By visualizing the process, a flowchart can quickly help to identify bottlenecks or inefficiencies. There are three basic types of flowcharts. The first type, lets call it basic flowchart, is used to outline a process quickly or to chart a process that involves few people. The second type, often called *opportunity flowchart* [88], is used to help to understand or improve a process that has many steps, including when things go right and when things go wrong. A *deployment flowchart* [87] illustrates the detailed steps in a procedure for each group of people involved in the process.

2.1.3 Look-Up Tables

Grid based look-up tables are, because of their simplicity, by far the most used models. Usually a set of observed input-output data is simply stored in a table and the model responds to an unseen input with the output calculated as a linear interpolation of the stored output values of the closest stored points to the actual input. A normal car produced at the beginning of the 21^{th} century contains about 100 grid based look-up tables. Grid based look-up tables are easy to implement models which has no need of parameter optimization. Due to the curse of dimensionality (see Sec. 3.3.4) this type of model is restricted to problems of very few inputs. Grid based look-up tables exhibit a strong similarity to mathematical models. In fact they can be interpreted as fuzzy models with triangular membership functions which fulfill the condition of complementarity (see Sec. 3.3.2).

2.1.4 Mathematical Models

The derivation of mathematical models are twofold. Firstly, mathematical models can be derived by the utilization of expert knowledge. Experts typically map their knowledge in an analytically expression by using differential equations or state space equations. This kind of model derivation is referred as *knowledge-based*, *theoretical*, *mechanistic*, *axiomatic* or *white-box modeling*. Knowledge-based modeling is only applicable if the real world process is fully understood. Because the resulting models are fully interpretable, they are called *white box models*. These theoretical derived models are widely used to model chemical, mechanical, electrical or fluid processes. Nowadays there exist big libraries containing whole model-packages and software to implement mainly time continuous models. Commonly used software packages are Dymola, Spice, Simpack, Hysis, AspernPlus, Adams and gProms.

Secondly, mathematical models can be derived by the utilization of available data. This kind of model derivation is called *data-driven*, *experimental*, *statistical* or *black-box*, modeling. The possible *analytical expressions* representing the model are various and the derived models are mainly used for control and prediction tasks. The interpretability of the resulting models depends on the chosen kind of analytical expression. The general characteristics of analytical expression are used to distinguish between so called *model classes*. Common mathematical model classes are, for example, artificial neural networks and fuzzy models. It is worth noting that this classification is very rough and that there exist several finer classifications, depending on the chosen model characteristics. If instances of the class of artificial neural networks are used to model a system, the resulting models are called *black-box models*, because artificial neural networks are mostly difficult to interpret. In the case of fuzzy rule based models, the resulting analytical expression is often referred as *white-box* model, because the system behavior can be easily formulated in human language and thus, is accessible to the human intellect. Furthermore there exist many hybrid forms utilizing more than one available concept and thus, there exist so called *grey-box* models (with various subdivisions) which can not be clearly classified as *black-box* or *white-box* models. Commonly used software packages for data-driven modeling are Matlab, Maple and Mathematica.

2.2 Application Areas of System Identification

The *application areas of system identification* can be classified by ”where” and ”what for” system identification is used. The ”where” is described by Fig. 2.1, which represents system identification usage of high frequency by deep black towards low frequency usage represented by a lighter gray. The decreasing frequency comes along with an increasing complexity and a decreasing expertise in the concerning application area.

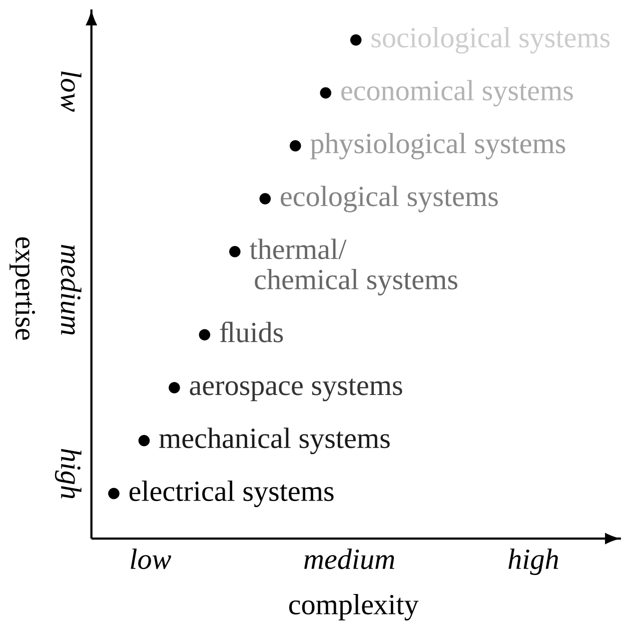


Figure 2.1: Application areas of system identification depending on expertise and complexity of the system. The darker the more common is the use of models.

The ”what for” can be divided in five main application areas, namely simulation, analysis, optimization, prediction and control. The next subsections give a short outline and some examples of models in these areas.

2.2.1 System Identification with Computational Intelligence

Although it should be evident, the author thinks that it is necessary to emphasize that this thesis does not cover the ”classical” methods of system identification. There exists excellent literature [185, 184, 100, 91] concerning the description of systems and their behavior, the mapping of knowledge into differential equations to describe real world systems and the application of filters to predict system states. Doing so and the related background knowledge is often subsumed by the term ”system theory” and the author strongly recommends to use and to apply this know-how if possible.

The aim of this thesis is to target systems, which firstly are too complex to be tackled with low parameterized models and secondly are too unknown to have the possibility to map human expertise into well formulated analytical expressions. The term system identification perfectly describes what is done in this thesis, namely mapping a real world process into a model. Because of this it can be realized that system identification or modeling is more and more done by methods which are classified as soft-computing methods [192, 4].

2.2.2 Simulation

Simulation is the classical field for system identification. If a reliable model of the target system is available, performing simulations can be traced back to one of the following reasons:

- The model provides a bigger specification range than the (implemented) system.
- Simulations are often cheaper than real world experiments.

The latter reason has to looked at from a financial as well as from a time consuming point of view. Widely known representatives are crash test models, production facility models or flight simulators.

2.2.3 Analysis

The most ambitious idea in model analysis is to use a data-driven model to get a deeper insight to the underlying real world system. A commonly used method to analyze data-driven models is to extract fuzzy rules from the model structure. A typical example of analysis based on data-driven models is *data mining*, where sometimes huge data bases are scanned for unknown relationships. Another interesting application are automatic theorem proofers. A list of research groups working in this field and available software can be found in [86].

More conservative approaches use knowledge-based models to play with some model functionalities to improve understanding of the functioning of the underlying process. A classical example are formulas, which can be seen as generalized models. In the domain of science the proving and arranging of formulas are based on varying and introducing model parameters.

2.2.4 Optimization

Optimization utilizes the model of a process to find optimal model inputs due to a desired process output. Often system outputs are contradictory and models are used to find a Pareto-optimal [85] set of parameters regarding the desired outputs. Obvious advantages to use models for optimization tasks are the saving of time and the decoupling from the real process. The precondition for optimizing is the availability of an accurate model for all operating conditions which may occur during optimization.

2.2.5 Prediction

For prediction models it is important to distinguish between open and closed systems. By modeling a closed loop system the possibility of predicting arbitrary many steps in the future exist. Note, that by using digital computers even the predictions of a perfect modeled system will become more and more inaccurate because of accumulating rounding errors. How fast this deviance grows depend on the *chaotic* behavior [161] of the system. By assuming a restricted, non-linear and deterministic dynamic system, deterministic chaos arises through positive and negative feedbacks. Positive feedbacks in form of local instabilities lead to a divergence in neighbouring values of system states. Globally appearing negative feedbacks have a stabilizing effect. If neither the positive nor the negative feedbacks get out of hand, the system stays in a limited space, following an aperiodic trajectory. This trajectory shows a sensitive dependency to infinitesimally small changes in the initial conditions of the system. Although very ”small” changes can lead after a ”short” time to totally different system behavior, the resulting trajectory is often self-similar. The behavior of a system that generates deterministic chaos can be explained by a deterministic non-linear (not necessarily known) model of differential equations.

If the modeled system is an open system, the prediction range is restricted by the last available input variable. Assume we want to model a system by using $u_1(t-7), u_2(t-4), u_2(t-6), u_3(t-5), y(t-1)$ and $y(t-3)$, where t denotes time-steps, u_1, u_2, u_3 denotes external system inputs and y denotes the model output. Furthermore assume that the latest available value is alway given by $t-1$ because most real processes have no direct feed-through. Under these terms the farest reaching prediction for this model with six inputs would be given by $y(t+3)$ (caused by $u_2(t-4)$), a so called 3-step-ahead prediction, or more general a k-step-ahead prediction with $k = 3$ as *prediction horizon*.

Prediction models are for example used for short-term stock market forecasts and weather forecast, but also for climatic progression forecast etc., which indicates that prediction is often utilized in optimization and analysis tasks. In fact, closed system modeling for prediction is synonymous to simulation.

2.2.6 Control

Most models for *control* tasks are implemented as look-up-tables [135]. As mentioned in Sec. 2.1.3 look-up-tables are the by far most used model type and because nearly all look-up-table based models are utilized in control tasks, control is the biggest application area for system identification.

2.3 Tasks in System Identification

The process of system identification includes three tasks, namely:

- Selecting a model class.
- Selecting the model structure.
- Parameter optimization of the model.

The selection of a model class could also be seen as a first stage in selecting a model structure. However, it is obvious that the space of all mathematical models comprises models with very different characteristics. In general a model designer has a couple of very specific characteristics in mind that a model should possess. Because of this, and to reduce the model search space, at the beginning of each modeling process a specific model class is selected. The latter two points are normally done iteratively, where the structure selection is done in boundaries specified by the model class, and the parameter optimization task is embedded into a loop of model structure selections.

2.3.1 Selecting a Model Class

The first and most fundamental task in data-driven modeling is the decision which model class should be used to model the target system. The framework of this model class will provide the basic conditions such as flexibility, interpretability and learning capability of the model. A model class used for data-driven modeling should meet the following conditions:

Universal approximation ability should be given. This is obvious since the objective of system identification is to model the target system as good as possible. An ex ante restriction in the modeling accuracy would be in conflict to most goals of system identification.

Availability of efficient learning algorithms should be given. This is a must for all data-driven approaches, since the model parameters has to be optimized on the basis of data.

Adjustable interpretability of the model class should be given, because a general approach to data-driven system identification should provide all application areas of system-identification reaching from control, where sometimes there is no need for interpretability, to analysis, where interpretability is indispensable. Since interpretability and accuracy are contradictory goals, the interpretability abilities of the model class should be adjustable.

Incorporation of expert knowledge should be possible. This point is not directly intelligible, because the modeling process is data-driven. But if the application area is for example analysis and the performed system identification was successful, the possibility to incorporate new insights obtained by model analysis should be given.

2.3.2 Selecting the Model Structure

Selecting a good model structure is the most challenging task in system identification. By considering data-driven system identification we assume that no expert knowledge about the real system is available. Thus, model structure optimization subsumes the task of identifying relevant inputs and the adaption of the internal connectivity structure of the model. Although there exist several methods to create model structures (see

Sec. 2.5.4 and 2.5.5), it has to be always in mind, that it is very restrictive to assume that the real system is in some special way decomposable (for example additively).

2.3.3 Parameter Optimization of the Model

Once a model structure is chosen, the model parameters[2] have to be adapted in such a way that the computed model output is as ”close” as possible to the desired system output. There exist two mainly used parameter optimization techniques for supervised learning. Firstly, the so called gradient descent methods like backpropagation [186, 156] which iteratively refine a solution and secondly, methods which directly solve a system of overdetermined equations. The latter concept will be used in this thesis and thus is discussed in greater detail in the next sections.

2.4 Parameter Optimization with Different Error Measures

In this thesis only *supervised learning* techniques are mentioned. All supervised learning methods are based on available knowledge about the input and output data of a process. The objective of such methods is to minimize some error measure, which is calculated by differences of the model behavior and the expected process output, in order to obtain an optimal model. The next subsections provide a mathematical expression of this error measure and based on this, methods to find (sub)optimal model parameters are shown.

2.4.1 Loss Functions and Cost Functions

Because in this thesis only single output models are considered, all following definitions and equations are formulated for this kind of models, e.g. the output of a model is written as a scalar. In order to optimize, the need of formulating a mathematical expression of what to optimize arises. Loss functions are used to measure the model output $\texttt{ModelOut}(\underline{u}) = \hat{y}$ of a single input vector $\underline{u}$ to a real valued error and cost functions are used to provide the analytical term which will be minimized to obtain a ”good” model. In supervised methods the value provided by the loss function is

[2]To be precisely the parameters P_1 of Eq. (2.21).

usually computed as the difference between the measured process output y_m and the model output $\texttt{ModelOut}(\underline{u}_m) = \hat{y}_m$, where $\underline{u}_m$ is the m^{th} input vector of a given set of input vectors (e.g. matrix $\mathbf{U}$), the so called training set. A formal definition of what is loss incurred by a model output $\hat{y}$ at location $\underline{u}$, given a desired y is given by the following definition.

Definition 2.1 (Loss function). *Let $(y,\hat{y}) \in \mathcal{Y} \times \mathcal{Y}$ be the tuple consisting of a the desired model output y and a calculated model output $\hat{y}$. Then a function $\mathcal{Y} \times \mathcal{Y} \rightarrow [0,\infty)$ with the property $\texttt{loss}(y,\hat{y}) = 0$ for all $y \in \mathcal{Y}$ will be called a loss function.*

Thus, a loss function defines a measure to assess a single model output and the so called *cost function* provides an expression to assess the model output for a set of inputs. A formal definition of what is cost incurred by a model output vector $\underline{\hat{y}}$ given a desired output vector $\underline{y}$ and an input matrix $\mathbf{U}$ can be characterized by definition.

Definition 2.2 (Cost function). *Let $(\texttt{loss}(\cdot),\mathbf{U}) \in \mathcal{L} \times \mathcal{U}$ be the pair consisting of an arbitrary loss function $\texttt{loss}(\cdot)$ and an $M \times N$ input matrix $\mathbf{U}$ consisting of M input vectors $\underline{u}_m$ of length N, the function $\texttt{cost} : \mathcal{L} \times \mathcal{U} \rightarrow [0,\infty)$ will be called a cost function.*

A common definition of a cost function is $\sum_{m=1}^{M} \texttt{loss}(\cdot)$, simply performing a summation of all losses caused by a set of input vectors.

2.4.1.1 Binary Classification

For binary classification the simplest loss function is given by

$$\texttt{loss}(y,\hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{otherwise.} \end{cases} \tag{2.1}$$

This definition does not distinguish between different classes nor between different types of error (i.e. *false positive* or *false negative*[3]). Replacing the ”otherwise” case in Eq. (2.1) by a function the incurred loss can be weighted.

This becomes necessary if the importance of the correctness of a model classification regarding different classes varies. For example a model has to classify blood

[3]A false negative is a pattern which the classifier wrongly assigns to class 1, a false negative is wrongly assigned to class -1.

donations into two classes, namely harmless (1) and contaminated (-1). The classification of a contaminated blood donation into the class harmless (false positive) has to be avoided at any price. On the other hand the misclassification of a harmless blood donation into the class contaminated (false negative) has only perishable consequences. Often it is necessary to take a certain confidence value for the classification result into account. In this case $\texttt{ModelOut}(\underline{u})$, which is used to calculate $\hat{y}$, becomes a real valued function, even though $y \in \{-1, 1\}$. In this case, $\text{sgn}(\hat{y})$ denotes the class label, and the absolute value $||\hat{y}||$ the confidence of the model output. Common corresponding loss functions are the ”soft margin” loss, the ”logistic” loss and the ”inverse complementary log-log” function. Matters become more complex when dealing with more than two classes. Because each type of misclassification could potentially incur different loss, $i \times i$ matrices, with i equal to the number of different classes, are used to store the possible confidence values.

2.4.1.2 Regression

The most common choice for loss functions dealing with real valued differences is

$$\texttt{loss}(y, \hat{y}) = (y - \hat{y})^2 \quad \text{or equivalently} \quad \hat{\texttt{loss}}(\xi) = \xi^2, \tag{2.2}$$

with ξ representing a tuple. For efficient implementation of learning procedures the loss function should be computationally cheap to evaluate. Furthermore it should have no or only a small number of discontinuities in the first derivative and it has to be convex in order to ensure the uniqueness of the solution.

The task of a learning procedure is to minimize the cost function. By using a loss function as given by Eq. (2.5) the most common cost function to be minimized is given by

$$\texttt{cost}(\xi^2, \mathbf{U}) = \sum_{m=1}^{M} (y - \hat{y})^2. \tag{2.3}$$

Linear optimization problems applying a cost function as given in equation (2.3) are called *least squares* (LS) problems. If Eq. (2.3) is used for nonlinear problem optimization the problem is called a *nonlinear least squares* problem. If furthermore the loss function is weighted, i.e.

$$\texttt{loss}(y, \hat{y}) = w(y - \hat{y})^2, \tag{2.4}$$

the concerning cost function offers the advantage that knowledge about the relevance and/or confidence of each data sample can be taken into account. Optimization problems minimized by cost functions using weighted loss functions are called *weighted least squares* and *weighted nonlinear least squares* problems, respectively.

Note, that a cost function as given in Eq. (2.3) is, because of it quadratic scaling of errors, very sensitive to outliers. By choosing as loss function the more general expression

$$\texttt{loss}(y,\hat{y}) = \|\, (y-\hat{y})\, \|^p \quad \text{or equivalently} \quad \widehat{\texttt{loss}}(\xi) = \xi^p \tag{2.5}$$

it is possible to show [133] that the more the exponent p rises the more the cost function is sensitive to outliers. This is the reason for another very common choice of the cost function, namely the sum of absolute errors, by choosing $p = 1$.

2.4.2 Linear Parameter Optimization

A problem whose model output $\hat{y}$ depends linearly on the N parameters w_n $(n = 1,\ldots,N)$ is referred as a linear optimization problem:

$$\hat{y} = w_0 + x_1 w_1 + x_2 w_2 + \cdots + x_n w_n = w_0 + \sum_{n=1}^{N} x_n w_n, \tag{2.6}$$

by, for sake the of simplicity, omitting the index m and the x_n can be (non-linear) transformed values of the original inputs u_n. In the following the parameters w_n will be called *weights*, the parameter w_0 will be called *bias* (*intercept-term* in the statistical jargon), y will be named desired model output and $\hat{y}$ will be denoted as computed model output. The usage of the term ”weights” for this kind of parameters has its origin in the artifical neural network community. In statistics the w_n are called *regression coefficients* or *parameter estimates*, the x_n are called *regressors* or *independent variable* and y is called the *dependent variable*.

2.4.2.1 Least Squares

As mentioned above the goal in LS is to minimize the concerning cost function. To derive an analytical solution we first rewrite the problem

$$\begin{array}{cccccccccccc}
w_0 & +x_{1,1}w_1 & + & x_{1,2}w_2 & + & \cdots & + & x_{1,n}w_n & = & y_1 \\
w_0 & +x_{2,1}w_1 & + & x_{2,2}w_2 & + & \cdots & + & x_{2,n}w_n & = & y_2 \\
\vdots & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\
w_0 & +x_{m,1}w_1 & + & x_{m,2}w_2 & + & \cdots & + & x_{m,n}w_n & = & y_m
\end{array}$$

in vector/matrix form

$$\mathbf{X}\underline{w} = \underline{y}, \tag{2.7}$$

with $\underline{w} = (w_0, w_2, \ldots, w_n)^{\mathrm{T}}$ and $\underline{y} = (y_1, y_2, \ldots, y_m)^{\mathrm{T}}$. In this case $\mathbf{X}$ is referred as the *regression matrix*. If $m \geq n$ the set is called *overdetermined* but since the Eq. (2.7) volitional represents an inadequate model of a real world problem, the existence of an exact solution is seldom given. In general a vector of residuals $\underline{r} = (r_1, r_2, \ldots, r_n)^{\mathrm{T}}$ with

$$\underline{r} = \mathbf{X}\underline{w} - \underline{y}, (\underline{r} \neq \underline{0}) \tag{2.8}$$

will remain. By utilization Eq. (2.3) as cost function we derive

$$\mathtt{cost}(\xi^2, \mathbf{X}) = \frac{1}{2}\underline{r}^T\underline{r} = \frac{1}{2}(\mathbf{X}\underline{w} - \underline{y})^T(\mathbf{X}\underline{w} - \underline{y}) \equiv MIN. \tag{2.9}$$

The norm $|| \, \underline{r} \, || = \sqrt{\underline{r}^T\underline{r}}$ of the residual vector is called *residuum*. Note that for convenience the cost function ist multiplied by $1/2$ in order to get rid of the factor 2 in the gradient. Considering Eq. (2.9), the gradient of the cost function with respect to the weight vector $\underline{w}$ has to be equal to zero. This leads to

$$\frac{\partial \mathtt{cost}(\xi^2, \mathbf{X})}{\partial \underline{w}} = \mathbf{X}^T\underline{r} = \mathbf{X}^T(\mathbf{X}\underline{w} - \underline{y}) = \underline{0} \tag{2.10}$$

or

$$(\mathbf{X}^T\mathbf{X})\underline{w} = (\mathbf{X}^T\underline{y}). \tag{2.11}$$

Equations (2.10/2.11) are called the *orthogonal equations* of the linear least squares problem, since at the optimum the residuals $\underline{r}$ are orthogonal to all regressors $\underline{x}_n$ (columns of $\mathbf{X}$). The transition from (2.7) to (2.10/2.11) is called *Gauss transfor-*

mation, because this proceeding was first described by Gauss in 1795. Note that $\mathbf{X}^T\mathbf{X}$ is identical to the *Hessian matrix*

$$\mathbf{H} = \frac{\partial^2 \texttt{cost}(\xi^2, \mathbf{X})}{\partial^2(\xi^2)} = \mathbf{X}^T\mathbf{X} \tag{2.12}$$

of the cost function. To compute the weight-vector $\underline{w}$ we have to solve

$$\underline{w} = (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\underline{y}), \tag{2.13}$$

which is denoted as the *least squares estimate*. The expression $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ is called the *pseudo inverse* of the regression matrix $\mathbf{X}$. An important fact is that the accuracy of a numerical inversion depends on the condition of the Hessian matrix and thus, on the condition of the regression matrix $\mathbf{X}$. The condition of a matrix can be defined by the ratio

$$\rho = \frac{\lambda_{max}}{\lambda_{min}}. \tag{2.14}$$

of the largest to the smallest eigenvalue of a matrix. Remember that a matrix is termed *orthogonal* if its transpose equals its inverse

$$\mathbf{X}^T = \mathbf{X}^{-1} \quad \text{or} \quad \mathbf{X}^T\mathbf{X} = \mathbf{X}\mathbf{X}^T = \mathbf{I} \tag{2.15}$$

and that a product of orthogonal matrices is also orthogonal. Thus, if the regression matrix is orthogonal so is the corresponding Hessian. By considering that the eigenvalues λ_k correspond to the variance of $\mathbf{X}$ projected to the k^{th} axis, an orthogonal Hessian with equal eigenvalues correspond to a contour plot of the cost function forming perfect circles and an origin identical to the origin of the weight space. The inversion of the Hessian with a numerical error equal to zero is possible. The more ρ rises and thus, the contour lines of the cost function becomes more elliptic, the lower is the numerically accuracy of the inversion. Therefore, a direct matrix inversion, with its bad numerical properties, is seldom performed. For ”big” residuals $||\underline{r}||$ and ”small” weights $||\underline{w}||$ often the very fast *Cholesky decomposition* [148] is used to build the pseudo inverse of $\mathbf{H}$. More stable approaches are *orthogonalization methods* [149] which base on the factorization of $\mathbf{X} = \mathbf{QR}$ where $\mathbf{Q}$ is a $M \times M$ matrix and $\mathbf{R}$ is a triangular matrix. A famous representative of this class is the *Householder transformation* [151], which is also used in this thesis.

2.4.2.2 Regularization

If $\lambda_{\min}$ reaches zero a matrix becomes singular (rank $\mathrm{Rk}(\mathbf{X}_{m,n}) \leq m$) and for this reason becomes uninvertible. If this happens with the Hessian matrix $\mathbf{X}^T\mathbf{X}$, a unique solution is no longer available, since for a rank deficit of one the solution is a line in the weight space, for a rank deficit of two the solution is a plane et cetera. A method to handle this uninvertibility is to apply *regularization* [152]. Furthermore regularization is used to improve the results obtained by the above described LS estimates. This improvement is caused by the fact that a good regularization leads to more ”circle-like” contour plots of the (hyper)parabolic cost function, with a minimum closer to the origin of the weight-space. Due to this fact and foremost to make inversion possible, the eigenvalues have to be changed. This can be done by adding a certain α to all diagonal elements of the Hessian $\mathbf{X}^T\mathbf{X}$, leading to

$$\underline{w} = (\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}^T\underline{y}. \tag{2.16}$$

Performing this regularization causes that zero eigenvalues are set to α and thus, the condition ρ of the Hessian matrix is no longer infinite. Considering the case of a ”very small” eigenvalue $\lambda_{\min}$, ρ at least decreases equal to a factor of $\frac{\alpha}{\lambda_{\min}}$. Metaphorically speaking the contour lines of the cost function become more ”circle-like” with a minimum closer to the origin of the weight-space. This approach is often denoted as *ridge regression* in statistics. Unfortunately there is a price which have to be paid. Firstly, the residual will increase as α increases, because only significant elements (with respect to their eigenvalue) of the regressors will contribute to calculate $\underline{w}$ and secondly, iterative search approaches to find (sub)optimal values for α has to be performed. Ridge regression is a *linear regularization* method and therefore a special case of the so called *Tikhonov-Phillips regularization*, which utilizes an arbitrary matrix $\mathbf{L}$ instead of the identity matrix $\mathbf{I}$ in Eq. (2.16).

In this thesis regularization only occurs in form of penalizing a fitness value during an evolutionary search process. Nevertheless, regularization schemes become very important if human expert knowledge should constrain a model [82]. In this case regularization restricts the flexibility (by preserving the complexity) of the regularized model.

2.4.3 Polynomial Models

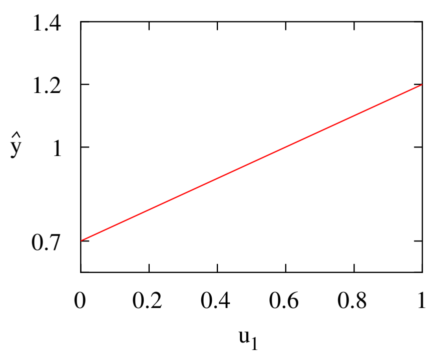
Polynomial models can be used to approximate linear and non-linear processes. If the available information about a process is very sparse and noisy, linear models are a good choice to describe this process. A *linear model* is a simple model with only a small number of parameters. Approximating a process of dimensionality N each linear model can be written as polynomial

$$\hat{y} = w_0 + w_1 u_1 + w_2 u_2 \ldots w_n u_n \tag{2.17}$$

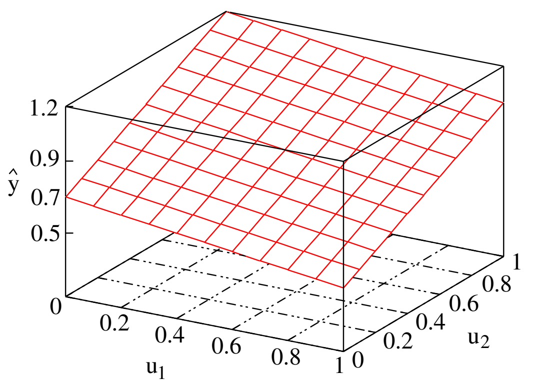
or more compact

$$\hat{y} = \sum_{n=0}^{N} w_n u_n \quad \text{with} \quad u_0 = 1, \tag{2.18}$$

and w_0 denoting the offset. Figure 2.2(a) shows a one-dimensional linear polynomial (order = 1), Fig. 2.2(b) shows a two-dimensional linear polynomial (order = 1).



(a) $\hat{y} = 0.7 + 0.5u_1$.



(b) $\hat{y} = 0.7 - 0.2u_1 + 0.5u_2$.

Figure 2.2: A one-dimensional polynomial of order one (a) and a two-dimensional polynomial of order one (b).

Linear models of higher dimensionality are represented by n-dimensional hyperplanes, which are graphically not presentable. The implementation of linear models is easy, the evaluation speed is fast, their sensitivity to noisy data is low, constraints of the model output can be incorporated by utilizing quadratic programming and prior knowledge can be utilized by applying regularization. For these reasons linear models are the standard models. Many systems can be approximated by linear models and to avoid breaking a butterfly on a wheel, linear models should be applied first. Not until the results of a linear system modeling are unusable, more parameters should be added to the analytical expression to perform a non-linear modeling. In the case of polynomials, non-linear approximation is done by using n-dimensional polynomials of order two or more. A *complete n-dimensional polynomial* of order k is given by

$$\hat{y} = w_0 + \sum_{n=1}^{N} w_n u_n + \sum_{n_1=1}^{N} \sum_{n_2=n_1}^{N} w_{n_1 n_2} u_{n_1 n_2} + \cdots + \sum_{n_1=1}^{N} \cdots \sum_{n_k=n_{k-1}}^{N} w_{n_1 \ldots n_k} u_{n_1 \ldots n_k}. \quad (2.19)$$

The offset together with the first sum describes a linear model. Each following sum increases the polynomial order by one. Thus, the second sum subsumes all quadratic terms of the polynomial (like $u_1^2, u_1 u_2, \ldots$), the third all cubic terms (like $u_1^3, u_1^2 u_2, u_1^2 u_3, u_1 u_2 u_3, \ldots$) and so on. The number of terms of a complete n-dimensional polynomial is equal to

$$P = \frac{(N+k)!}{N!k!}. \quad (2.20)$$

By using non-linear models it should always be in mind, that the number of model parameters no longer rise linearly but exponentially. Therefore some complexity consideration as introduced in the following sections should be known.

2.5 Model Complexity and Regularization

Model complexity considerations are independent from specific properties such as whether models are linear or non-linear. This section explains the bias/variance dilemma and the therewith aligned terms *overfitting* and *underfitting*. Based on these considerations the importance of different datasets for system identification and system validation is demonstrated. In this context some possible proceedings are presented when dealing with very small datasets. Finally some modeling approaches and templates of system structures are introduced that can help to reduce the complexity of the modeling problem and the finally identified system structure, respectively.

2.5.1 Model Complexity and Model Flexibility

In the mid 1960s Kolmogorov defined the *algorithmic complexity* [102] of a given function to be the length of the shortest model that describes the function after a finite amount of computation. Thus, *Kolmogorov complexity* [109] is an expression that can be used as a neutral reference point to measure the complexity of mathematical models. Because in this thesis only algorithmically describable models are considered, the *complexity of a model* is defined as

$$N = P_1 + P_2, \tag{2.21}$$

where P_1 is the total number of operators and P_2 is the total number of operands. In the following complexity and number of parameters are used as synonyms. Furthermore, the estimation of ”best” values for P_1 will be referred to as *parameter optimization* and the estimation of ”best” values for P_2 will be denoted as *structure optimization*.

It is important to notice that each parameter has not necessarily the same impact on the coverage of possible state spaces of the system. Let us define *flexibility* as the value of the accessible state space of a model by performing parameter variation. If a model possesses parameters which have no influence on the model flexibility, for example, when applying regularization, the remaining parameters which affects the flexibility are referred to as *effective parameters*. Thus, the parameters P_1 of a model determine the search space for parameter optimization and the effective parameters determine the accessible state space of a model. Logically, complexity and flexibility should not be used as synonyms, because a more complex model is not necessarily as flexible as a model with fewer parameters (see section 2.5.5).

2.5.2 Bias Error and Variance Error

Each model error can be decomposed in two different parts, namely the *bias* error and the *variance* error. Assuming Eq. (2.5) as loss function we can write

$$\begin{aligned} E(\cdot) &= E\left((y-\hat{y})^2\right) \\ &= E\left((\tilde{y}+n-\hat{y})^2\right) \\ &= E\left((\tilde{y}-\hat{y})^2\right) + E(n^2), \end{aligned} \tag{2.22}$$

$$\begin{aligned} \text{with} \quad y &: \text{ measurable process output,} \\ \tilde{y} &: \text{ noisefree process output,} \\ \hat{y} &: \text{ model output,} \\ n &: \text{ noise.} \end{aligned}$$

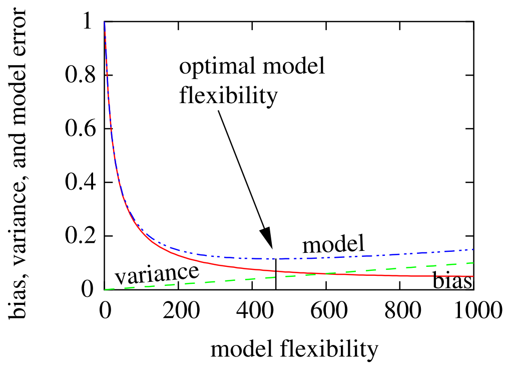
Obviously Eq. (2.22) splits the measured process output into the unmeasurable true process output and the noise variance. The loss function is minimal if the model describes the true process perfectly (i.e. $\hat{y} = y$). Thus, the loss function value becomes equal to the noise variance. Because the model does not influence the noise variance, only the decomposition of the model error y - $\hat{y}$ is considered in the following.

$$\begin{aligned} E\left((y-\hat{y})^2\right) &= E\left((y-E(\hat{y})-(\hat{y}-E(\hat{y})))^2\right) \\ &= E\left((y-E(\hat{y}))^2\right) + E\left((\hat{y}-E(\hat{y}))^2\right) \\ &= (y-E(\hat{y}))^2 + E\left((\hat{y}-E(\hat{y}))^2\right) \\ (\text{model error})^2 &= (\text{bias error})^2 + \text{variance error} \end{aligned} \tag{2.23}$$

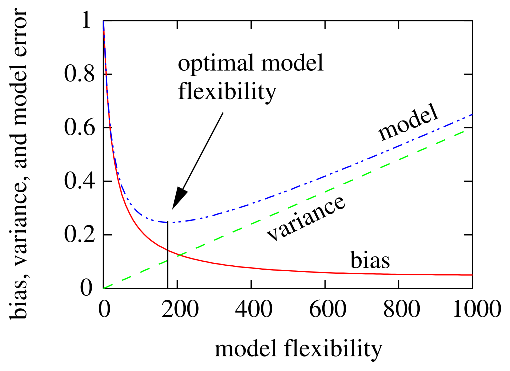
If the chosen model structure is flexible enough, the parameters of the model can be set to optimal values and the bias error will be zero. This is, for example, the case for linear models of order n (i.e. with $2 \cdot n$ parameters) which are used to model linear processes of less or equal order $m \leq n$. On the other hand, if $m > n$ the model structure is not flexible enough to model the process exactly and the error due to this process/model mismatch is called bias error. To come out with a zero bias error by modeling arbitrary[4] nonlinear processes, so called *universal approximators* are needed. For this kind of approximator models (e.g. polynomial, artifical neural networks, fuzzy systems, etc.) a zero bias error can always be achieved by increasing the complexity of the approximator structure.

2.5.3 Bias/Variance Tradeoff

Obviously a too simple model has a high bias error, since it can not predict the noise-free system states, but a low variance error. On the other hand a too complex model has a low bias, but a high variance error. A too simple model can be improved by adding parameters because, the increase in the variance error is overcompensated by a decrease in the bias error. On the other hand, a too flexible model can be improved by discarding parameters because, the increase in the bias error is overcompensated by a decrease of the variance error. Thus, the optimal model, on the basis of certain data,



(a) Low variance error due to less noise or more data.



(b) High variance error due to high noise or few data.

Figure 2.3: Bias/variance tradeoff. The optimal model flexibility is determined by the model error which can be decomposed into a bias and a variance part.

is a model somewhere between and the contradictory behavior of bias and variance error is referred as bias/variance tradeoff or *bias/variance dilemma*. Figure 2.3 illustrates the effect of different bias and variance errors on the model error. Unfortunately

[4]Not total arbitrary because only smooth processes are considered.

bias and variance errors are unknown in practice. Thus, one of the two possible techniques, as described in section 2.5.4 and section 2.5.5, has to be applied to identify a (sub)optimal model flexibility.

2.5.4 Implicit Structure Optimization

Implicit structure optimization also referred as *regularization* is often used if the estimation of the model error is computationally expensive. Regularization techniques decrease the *model flexibility* by retaining the complexity of the model. Logically, regularization is only applicable to already overly flexible models. Because of this restriction implicit structure optimization is used in this thesis only for fitness penalization and not in the classical context. The interested reader is referred to [134] which provides a good overview to regularization techniques.

2.5.5 Explicit Structure Optimization

Explicit structure optimization is mostly used if the estimation of the model error is computationally cheap, because then it becomes possible to evaluate several models with different number of parameters. These models are compared by their resulting errors computed on the test dataset. Explicit structure optimization can be divided into the following four categories:

Forward selection is a strategy which starts with a very simple or empty model and gradually increases the flexibility of this model by adding either new parameters or whole substructures. At each iteration step, a number of possible ways in which the model can be made more flexible is identified and the according model errors are computed. The optimal refinement step is selected and included in the current model. This is done until the models performance is acceptable or become worse. The advantage of using forward selection is that unnecessarily complex models do not have to be computed. Representatives of model classes which are optimized by the usage of forward selection are:

- Linear parameterized models which can utilize orthogonal least squares [22, 23] algorithms.

- The projection pursuit [55] algorithm which build up multilayer perceptron neural networks with individually activation functions by using a staggered optimization process to adapt the weights.
- The classical artificial neural networks, which than often refer to the term *growing networks* like Marchand's algorithm [117], tiling algorithm [130], upstart algorithm [54], cascade-correlation algorithm [51] or the scheme of simple expanding recurrent neural networks [24].
- Tree based approaches [56, 153, 163] like the *local linear model tree* (LOLIMOT) [131, 132] which iteratively partitions the input space.
- The *adaptive spline modeling* (ASMOD) [97] algorithm which assumes that the desired system behavior can be additively decomposed, such that it can be successfully modeled from a linear combination of several n-dimensional sub-models.

Backward elimination starts with a very complex model and iteratively deletes parameters or substructures. Representatives of model classes which are optimized by the usage of backward elimination are:

- Linear parameterized models which can use an orthogonal least squares.
- The classical artificial neural networks, which than often refer to the term *pruning*. A survey can be found in [155].

Stepwise selection is a mixture of forward selection and backward elimination. In general all forward selection methods can be extended by backward elimination, which is normally performed in order to discover and discard redundant parameters or substructures. If forward selection and backward elimination is both considered in each iteration, this is referred as stepwise selection with the classification and regression tree [15] (CART) and multivariate adaptive regression splines [56] (MARS) as typical representatives.

Evolutionary based structure optimization is a general method to identify optimal bias/variance error balanced models. Obviously the comparison of all models with different flexibility would lead to the optimal model. Unfortunately this approach becomes even for small problems infeasible, since the search space is enormous. A possible way to handle these enormous search space is to use evolutionary computation (EC), which have been theoretically and empirically proven to provide the means for efficient search, even in complex spaces [61]. Thus, EC has become a common and general method for structure optimization. This approach was also chosen in this thesis and will be discussed in greater detail in chapter 5.

2.6 Model Generalization Estimation

The generalization error is the model error emerged from unseen data. In this thesis generalization estimations are persistently used to select an optimal model (regarding to a given cost function) out of a set of models. Normally generalization estimations are tackled by statistical methods, which give assumptions how statistics asymptotically distribute by increasing average sample size. But if some of the premises, as for example the assumption of normal distribution of certain variables, are violated or if only very few data samples are available, the asymptotical behavior can not be guaranteed and the consequences are disputable. To avoid this kind of problems this section focuses only on methods which can approximate the distribution of parameters without any strong assumptions.

As mentioned above many problems do not provide enough data to calculate a reliable generalization error of a final chosen model. Thus, model selection algorithms have to use error estimations or other model dependent information criteria. Section 2.6.2 gives an insight to generalization estimation techniques which can be used if the available dataset is sufficient. In Sec. 2.6.3 generalization estimation techniques on small datasets will be discussed and Sec. 2.6.4 gives a short outline of alternative usable information about the model.

2.6.1 Good and Best Feature-Subset

After choosing a final model, the parameter optimization of this model is always performed by using the whole dataset with M samples. In the following we will refer to $M_{\text{train}} = M - j$ ($j \ll M$ is the fold size of the cross-validation) as the dataset used for each validation, to a *good feature-subset* as a subset which contains all relevant inputs of the matrix $\mathbf{U}$ and to the unique *best feature-subset* that contains all relevant inputs but no others.

2.6.2 Training-, Validation- and Test-Dataset

Considering a three times representative dataset, this means each input space region is covered at least three times by very similar data, a common approach is to divide the dataset in training, validation and test data. The *training dataset* is used to optimize the parameters of a chosen model. The parameter optimized model is validated by computing its error on the *validation dataset*. Thus, if model selection should be performed, the validation dataset is used to choose a model from a set of available models and consequently the validation dataset is responsible for structure optimization in the model identification process. Finally, after structure and parameter optimization, the identified model is tested with unseen data, the so called *test dataset*. This whole procedure of optimizing and evaluating the generalization error is referred to as *split sampling*. Unfortunately split sampling is only applicable for at least three times representative data, which is for most real world problems not available. If this constraint can not be fulfilled, the need for computationally more expensive validation approaches arise [65].

2.6.3 Cross-Validation

In *j-fold cross-validation* (sometimes called *rotation estimation*), the data is randomly divided into j disjoint subsets of (approximately) equal size. The model is trained j times, each time leaving out one of the subsets from training, but using only the omitted subset with k members to compute the chosen error criterion. The mean of the evaluated j model errors is the overall j-fold cross validated model error (E_{cv}). A first formal description of j-fold cross-validation was given by [172] in 1974. If j equals the total available sample size M, this is called *leave-one-out cross-validation* . The leave-one-out approach is the computationally cheapest representative in the class of *complete cross-validation* techniques. All members of complete cross-validation techniques use as accuracy measure the average of all $\binom{M}{M/j}$ possibilities for choosing M/j instances out of M. Thus, repeating j-fold cross-validation multiple times using different splits provides a better Monte-Carlo estimate to the concerning *leave-j-out* model error.

2.6.3.1 Cross-Validation with Discontinuous Loss Functions

For model error estimation of continuous cost functions the use of leave-one-out cross-validation often works well, but if the cost function is discontinuous, for example in the case of binary classification, leave-one-out cross-validation may perform poorly and j-fold cross-validation should be preferred. A common choice is $j = 10$, because if j gets too small, the error estimate becomes pessimistically biased because of the difference in the dataset size between the full-sample analysis and the cross-validation analyses.

2.6.3.2 Cross-Validation vs. Complete Cross-Validation

Foremost it is notable that leave-one-out cross-validation for model selection is often a bad choice, because many problems possess the property that small changes in the data causes large changes in the model selected [17]. Considering the selection of feature-subsets in linear regression, 5-fold and 10-fold cross-validation works better than leave-one-out [16]. Even values of j in the range of $j = 2$ to $j = 4$ may work better if the j-fold cross-validation is done repeatedly to refine the model error estimate towards the concerning leave-j-out cross-validation.

2.6.3.3 Equivalence of Leave-j-Out to Information Criteria

It was shown that leave-one-out cross-validation using deviance as loss function is asymptotically equivalent to *Akaike's information criterion* (AIC) [173] (for information criteria see also Sec. 2.6.4), but leave-j-out ($j > 1$) cross-validation is asymptotically equivalent to the *Bayesian information criterion* (BIC) if the following condition holds [166]:

$$j = M\frac{1}{(\log M - 1)}. \tag{2.24}$$

BIC will choose the *best feature-subset* with probability $\lim_{M\to\infty} P = 1$, whereas AIC will choose only a *good feature-subset* with an asymptotic probability of one [175]. Furthermore other studies [78, 166] have found that AIC overfits badly in small samples where BIC works well.

2.6.3.4 Feature-Subset Selection

A notable observation is that for selecting feature-subsets by leave-j-out $(j > 1)$ in a linear regression the probability of selecting the best feature-subset is $\lim_{M\to\infty} P \neq 1$, unless $\frac{j}{M} = 1$ [165]. To give a better understanding of this statement, recall that by omitting the noise variance the model error consist of a variance error and the squared bias error as given in Eq. (2.23). By assuming a linear function to be learned, the bias for ”good” feature-subsets is zero, thus, the generalization error of good subsets in linear problems will differ only in the variance error

$$E\left((\hat{y} - E(\hat{y}))^2\right) = \frac{2p}{M_{\text{train}}}, \tag{2.25}$$

with p as the number of inputs in the feature-subsets. By definition the ”best” feature-subset has the smallest value of p. If M tends to infinity the differences in the model error among the models with a good feature-subset will all go to zero. Therefore it is difficult to guess which subset is best based on the model error, even if M is very large. It is well known that unbiased estimates of the model error, such as those based on AIC (see 2.6.4), do not produce consistent estimates of the best subset [175]. In leave-j-out cross-validation M_{train} is equal to $M - j$ and thus, the differences of the cross-validation estimates of the model error among the good feature-subsets contain a factor $\frac{1}{M_{\text{train}}}$. By making M_{train} small enough (and thereby making each regression based on M_{train} cases bad enough), we can make the differences of the cross-validation estimates large enough to detect. It turns out that to make M_{train} small enough to guess the best subset consistently, we have to have $\lim_{M\to\infty} \frac{M_{\text{train}}}{M} = 0$. The crucial distinction between cross-validation and split-sample validation is that with cross-validation, after guessing the best subset, the training of the linear regression model for that subset is done by using all M cases. In split-sample validation only M_{train} cases are ever used for training. If the main purpose is really to choose the best subset, it can be suspected to have $\frac{M_{\text{train}}}{M}$ go to zero even for split-sample validation. But choosing the best subset is not the same thing as getting the best model error. If the interest is more to achieve a good generalization than in choosing the best subset, it is not favorable to base the regression estimate in cross-vaidation on only M_{train} cases, because in split-sample validation that bad regression estimate is what we are stuck with. Thus, there is a conflict between the two goals of choosing the best subset and getting the best generalization in split-sample validation.

2.6.3.5 Bootstrapping

Bootstrapping (sometimes referred as *bagging*) was proposed by [47, 48] and a good introduction can be found in [129]. The bootstrap method uses the original dataset $\mathbf{D}^{\text{org}}$, consisting of M data-pairs $(\underline{x}_m, y_m)$, to generate any number of synthetic datasets $\mathbf{D}_1^{\text{syn}}, \mathbf{D}_2^{\text{syn}}, \ldots,$ also with M samples. The differences to the original dataset is, that the synthetic datasets can have multiple instances of one data-pair, because the data-pairs are drawn with replacement from $\mathbf{D}^{\text{org}}$. Typically each synthetic datasets has a random fraction of $\sim 1/e \approx 37\%$ duplicated instances. The synthetic datasets are now used for parameter optimization, yielding in a set of simulated measured weight vectors $\underline{w}_1^{\text{syn}}, \underline{w}_2^{\text{syn}}, \ldots,$ which will be distributed around $\underline{w}^{\text{org}}$. The weight-vector $\underline{w}^{\text{org}}$ again is considered to be distributed around $\underline{w}^{\text{true}}$. For a large class of problems the bootstrap method does yield easy and very quick Monte Carlo estimates of the real generalization error. Good results were achieved in the field of artificial neural networks [120, 181]. In contrast, bootstrapping seems to perform bad on empirical decision trees [101]. Always have in mind that bootstrapping supposes that the dataset $\mathbf{D}^{\text{org}}$ consist of M independent and identically distributed data points. Thus, in independent and identically distributed data the sequential order of data points is not of consequence to the process that is used to compute $\underline{w}$.

2.6.4 Information Criteria

A possibility to avoid computational cost during validation is the usage of *information criteria* (IC) instead of error estimates based on the repetitive model parameter recalculation on subsets of the available data. All information criteria are of the form

$$\text{information criteria} = \text{measure of fit} + \text{complexity penalty} \tag{2.26}$$

and the best model is defined as the model with the lowest information criterion. Thus, parameter optimization is still done by minimizing a cost function $\texttt{cost}(\cdot)$, but model selection is performed by comparing the information criteria of the different models. A demand on all *complexity penalties* is that they should increase with the number of model parameters N and should decrease while the number of data M increases. Furthermore, in the limit $M \to \infty$ the complexity penalty should tend to zero because the variance error also tends to zero. Two common information criteria are:

- *Akaike's information criterion* (AIC)[5] :
  AIC(ρ) = $M \ln(\texttt{cost}(\cdot)) + \rho N$ (with $\rho = 2$ as the most common choice).

- *Bayesian information criterion* (BIC) :
  BIC = $M \ln(\texttt{cost}(\cdot)) + \ln(M)N$.

In the case of applying a regularization method to a model and model selection is done by IC, remember (see Sec. 2.5.1) that the effective number of parameters N_{eff} of the model decrease and N has to be replaced by N_{eff}.

2.7 Summary

This chapter introduced the basic concepts which are necessary to understand the problems related with system identification. After a brief description of different model types a list of possible application areas was given. A more detailed view to system identification was emerged by describing the tasks to be performed for every system identification process. The possibilities to optimize parameters in a fixed model structure were discussed and several approaches to achieve a model structure were presented. The least squares parameter optimization method was mathematically inspected and will later be used to optimize the parameters of Takagi-Sugeno fuzzy models. The important subject of model complexity respectively flexibility and the strongly related subject of model generalization estimation were discussed in greater detail to provide the necessary background to model validation schemes.

[5]In fact AIC is the acronym for "An Information Criterion" as originally suggested from Akaike.

Chapter 3

Takagi-Sugeno Fuzzy Models

3.1 Fuzzy Logic and Fuzzy Models

Fuzzy logic is an extension to the classical two-valued logic by concepts of fuzzy set theory, introduced by Zadeh in 1965 [194]. A *fuzzy model* (FM) makes use of fuzzy logic concepts to represent a knowledge-base and/or to model interactions and relations of system variables. The application of fuzzy logic to rule-based models leads to the class of fuzzy rule based models which consider ”if-then” rules whose antecedents and, model dependent, consequents are composed of fuzzy statements. In the following, unless otherwise stated, the term fuzzy model is used in the meaning of fuzzy rule based model. Knowledge in FMs is represented by *linguistic variables* with an associated set of *linguistic values*. Linguistic values are defined by fuzzy sets, where a *fuzzy set* A in U is a set $A = \{u, \mu_A(u) | u \in U\}$ of ordered pairs which are defined by a *membership function* (MF) $\mu_A(u) \in [0,1]$ and a *linguistic term* for labeling.

Lin and Lee [111] excellently remarked that: ”One of the biggest differences between crisp and fuzzy sets is that the former always have unique MFs, whereas every fuzzy set has an infinite number of MFs that may represent it”. In a broad sense any field can be fuzzified and hence generalized by replacing the concept of a crisp set in the target field by the concept of a fuzzy set. Examples for basic fields which can be fuzzified are graph theory, arithmetic or probability theory resulting in fuzzy graph theory, fuzzy arithmetic and fuzzy probability theory. Examples for applied fields are stability theory, neural networks or mathematical programming resulting in fuzzy stability theory, fuzzy neural networks and fuzzy mathematical programming.

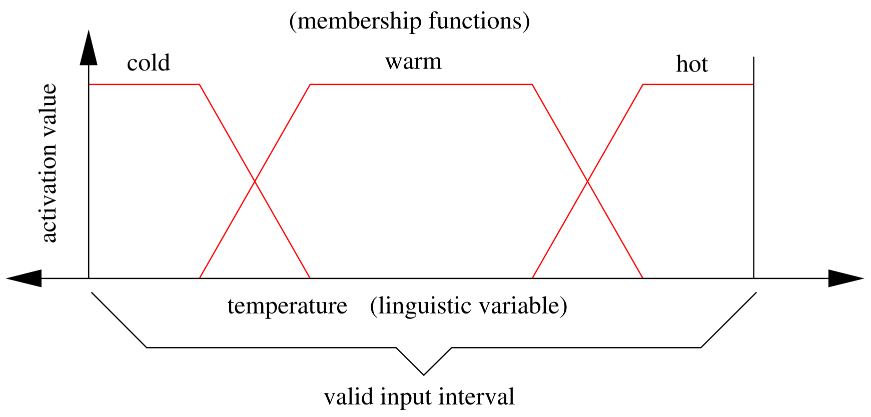


Figure 3.1: A linguistic variable named temperature with assigned fuzzy sets.

Fuzziness should not be confused with probability since it deals with deterministic plausibility, while probability concerns the likelihood of nondeterministic, stochastic events. Fuzziness and randomness differ in nature because they represent different aspects of uncertainty. The uncertainty of fuzziness is found in the definition of a concept or the meaning of a term such as ”cold water” or ”old person”, whereas the uncertainty of probability generally relates to the occurrence of phenomena. From a modeling point of view fuzzy MFs represent similarities of objects to imprecisely defined properties, while probabilities give information about relative frequencies.

3.2 Fuzzy Inference Systems

A *fuzzy inference system* is a rule-based model that uses fuzzy logic to reason about data [191]. The relationships between system variables are represented by means of fuzzy if-then rules of the following form:

IF antecedent proposition **THEN** consequent proposition.

The antecedent proposition is always a fuzzy proposition of the type ”u_n is A” where u_n is the n^{th} element of the model input vector $\underline{u}$ and A is a fuzzy set. The proposition‘s truth value depends on the degree of similarity between u_n and A.

The basic structure of a fuzzy inference system consists of four main components (see Fig. 3.2), namely:

A **fuzzifier** which maps crisp (real-valued) inputs into fuzzy values.

An **inference engine** that applies a fuzzy reasoning mechanism to obtain a fuzzy output.

A **defuzzifier**, which maps the fuzzy output into a crisp (real-valued) output.

A **knowledge-base** which contains both a set of fuzzy rules, known as the *rule-base*, and a set of MFs known as the *database*.

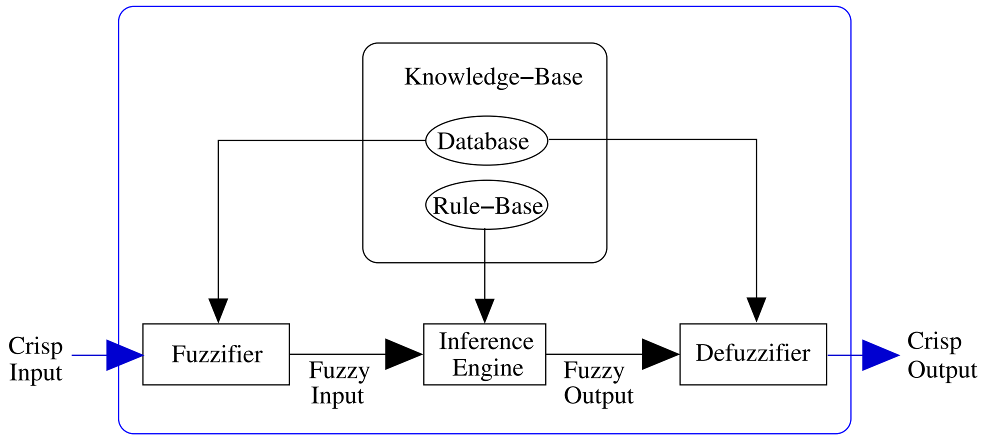


Figure 3.2: Basic structure of a fuzzy inference system.

All fuzzy rule-based models share this general structure. A categorization of FMs into the three mostly mentioned model classes depends on the form of the consequent of the fuzzy if-then rules and is given as follows:

In **linguistic fuzzy models** both, the antecedent and the consequent, are fuzzy propositions.

Fuzzy relational models are generalizations of linguistic models in which the relation between antecedent and consequent terms is fuzzy.

In **Takagi-Sugeno fuzzy models** the antecedent is a fuzzy proposition and the consequent is a crisp function.

More general, the classification of a fuzzy rule-based model is based on different assignments to logical parameters of the model, where the model parameters can be classified into four distinct categories [142] (Tab. 3.1):

Class	Parameters
Logical	MF types Fuzzy operators Reasoning mechanism Defuzzification method
Structural	Relevant features Number of MFs Number of rules
Connective	Antecedents of rules Consequents of rules Rule weights
Operational	MFs values

Table 3.1: Parameter classification of a fuzzy inference system.

The logical parameters are usually predefined by the designer based on available software tools and/or on the general problem characteristics. This thesis focuses on Takagi-Sugeno FMs because of the problem characteristics of data-driven modeling. Thus, the next sections provide a description of the logical parameters chosen to perform data-driven system identification, why they were chosen and where these models are situated in the soft-computing nomenclature.

3.2.1 Membership Function Types

The *support* of a MF is the crisp set of all points u in U which fulfill $\mu_A(u) > 0$. A fuzzy set whose support is a single point in U with $\mu_A(u) = 1$ is called a *fuzzy singleton*. Commonly used MFs are triangular, trapezoidal, bell-shaped and Gaussian functions. In this thesis b-splines are used as MFs because they offer some interesting characteristics. *B-splines* are defined over a knot-vector $\underline{\lambda}$, consisting of at least $k+1$ elements, with k denoting the order of the b-splines. Each element of the knot-vector is called knot and the b-spline values can be recursively [32, 29] calculated by

$$B_j^{k+1}(u) = \frac{u-\lambda_j}{\lambda_{j+k-1}-\lambda_j} B_j^k(u) + \frac{\lambda_{j+k}-u}{\lambda_{j+k}-\lambda_{j+1}} B_{j+1}^k(u), \text{ where}$$

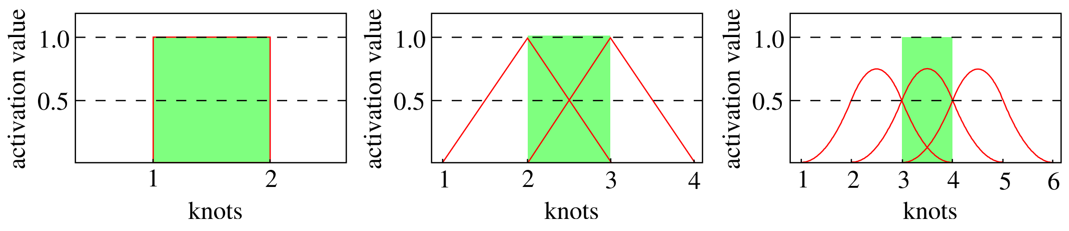
$$\begin{aligned} B_j^1(u) &= 1, \text{if } u \in [\lambda_j, \lambda_{j+1}) \\ &= 0, \text{otherwise.} \end{aligned} \tag{3.1}$$

Abbreviation:	u :	input value,
	$B_j^k(u)$:	activation value of the j^{th} b-spline defined over the knots λ_j to $j+k$.

The concept of a knot-vector and the characteristics of b-splines meets our demands in four ways.

- It is compatible with the construction demands of so called descriptive FMs. Each single knot-vector defines a whole set of MFs for one linguistic variable. These sets of MFs define, together with a labeling, the globally interpretable fuzzy sets which cover the input space.
- It is easy to achieve a finer grid of linguistic terms on each one-dimensional projection by simply inserting some knots in the concerning knot-vector.
- B-splines can form extremely different shapes simply by changing their order or their knot-positions.
- B-splines show some for FMs essential characteristics such as positivity and local support and furthermore they form a partition of unity (activation of all b-splines defined by one knot-vector sum up to the same value) which simplifies the interpretation of a FM and improves its learning capabilities [187].

Obviously (see Fig. 3.3(c)) b-splines of higher order ($k > 2$) are no longer normalized to one. From a semantic point of view this is no problem. In fact, single MFs with a maximum activation below one, by simultaneously fulfilling the partition of unity, normally are better qualified to fit the ideas behind fuzzy reasoning. This statement will be justified in Sec.3.3.2.



(b) Order 2. (c) Order 3.

Figure 3.3: Univariate b-spline functions of different order. The green shaded areas highlight the domain where the partition of unity is valid.

3.2.2 Fuzzy Operators

As with classic sets we need to define operations on fuzzy sets because we would like to be able to use compounds of linguistic descriptions (i.e. rules) in a mathematical way. The intersection and union operations of fuzzy sets, which are often referred to as *t-norms* (triangular norms) and *t-conorms* (triangular conorms), respectively [42, 43], are used to aggregate rule antecedents and to calculate the rule consequents. T-norms are two-parameter functions of the form

$$t : [0,1] \times [0,1] \rightarrow [0,1], \tag{3.2}$$

such that

$$\mu_{A\cap B}(u) = t[\mu_A(u), \mu_B(u)], \tag{3.3}$$

where the function $t(\cdot,\cdot)$ satisfies the following conditions:

Boundary conditions:	$t(0,0)=0$,
	$t(\mu_A(u),1)=t(1,\mu_A(u))=\mu_A(u)$.
Commutativity:	$t(\mu_A(u),\mu_B(u))=t(\mu_B(u),\mu_A(u))$.
Monotonicity:	If $\mu_A(u)\le\mu_C(u)$ and $\mu_B(u)\le\mu_D(u)$
	then $t(\mu_A(u),\mu_B(u))\le t(\mu_C(u),\mu_D(u))$.
Associativity:	$t(\mu_A(u),t(\mu_B(u),\mu_C(u)))=t(t(\mu_A(u),\mu_B(u)),\mu_C(u))$.

Typical nonparametric t-norms are:

Intersection:	$a\wedge b$	=	$\mathtt{min}\,(a,b)$.
Algebraic product:	$a\cdot b$	=	ab.
Bounded product:	$a\odot b$	=	$\mathtt{max}\,(0,a+b-1)$.
Drastic product:	$a\hat{\odot}b$	=	$\begin{cases} a, & b=1\\ b, & a=1\\ 0, & a,b<1.\end{cases}$

T-conorms (often referred as s-norms) are two-parameter functions of the form

$$s:[0,1]\times[0,1]\rightarrow[0,1], \tag{3.4}$$

such that

$$\mu_{A\cup B}(u)=s[\mu_A(u),\mu_B(u)], \tag{3.5}$$

where the function $s(\cdot,\cdot)$ satisfies the following conditions:

Boundary conditions:	$s(1,1)=1$,
	$s(\mu_A(u),0)=s(0,\mu_A(u))=\mu_A(u)$.
Commutativity:	$s(\mu_A(u),\mu_B(u))=s(\mu_B(u),\mu_A(u))$.
Monotonicity:	If $\mu_A(u)\le\mu_C(u)$ and $\mu_B(u)\le\mu_D(u)$
	then $s(\mu_A(u),\mu_B(u))\le s(\mu_C(u),\mu_D(u))$.
Associativity:	$s(\mu_A(u),t(\mu_B(u),\mu_C(u)))=s(s(\mu_A(u),\mu_B(u)),\mu_C(u))$.

Typical nonparametric t-conorms are:

Union:	$a \vee b$	$=$	$\mathtt{max}(a,b)$.
Algebraic sum:	$a \hat{+} b$	$=$	$a+b-ab$.
Bounded sum:	$a \oplus b$	$=$	$\mathtt{min}(1,a+b)$.
Drastic sum:	$a \hat{\oplus} b$	$=$	$\begin{cases} a, & b=0 \\ b, & a=0 \\ 1, & a,b>0. \end{cases}$

There exist several parametric t-norms and t-conorms [111] with the standard min and max operations as, respectively, the upper bound of t-norms (the weakest intersection) and the lower bound of t-conorms (the strongest union). The t-norms and t-conorms can be seen as aggregation operations on fuzzy sets, which combine several fuzzy sets to produce a single set. A general definition for an aggregation operation was formulated by [99] in the following way:

$$h : [0,1]^n \rightarrow [0,1], \quad n \geq 2, \tag{3.6}$$

such that

$$\hat{\mu}_A(u) = h(\mu_{A_1}(u), \mu_{A_2}(u), \ldots, \mu_{A_n}(u)) \quad \forall u \in U. \tag{3.7}$$

3.2.3 Reasoning Mechanism

There exist four principal modes of fuzzy reasoning, namely *categorical reasoning, syllogistic reasoning, dispositional reasoning* and *qualitative reasoning*. Qualitative reasoning refers to a mode of reasoning in which the antecedents and/or consequents propositions involve fuzzy or linguistic variables. Because qualitative reasoning plays a key role in fuzzy logic applications in the realms of control and system analysis [116, 176, 145, 108, 21], this thesis will focus on this mode of reasoning. For information about the other modes of reasoning the reader is referred to [112, 195].

3.2.4 Defuzzification Method

If the used consequents propositions involve fuzzy or linguistic variables, the model output has to be defuzzified in order to obtain one crisp output value. Obviously this is not necessary if the model output is presented to another fuzzy model or directly to a human. By using Takagi-Sugeno FMs the defuzzification step is omitted, since the consequents propositions of rules in Takagi-Sugeno FMs does not contain fuzzy or linguistic variables.

3.2.5 The Output Evaluation of a Takagi-Sugeno Fuzzy Model

To consolidate the above discussed concepts the principle procedure to evaluate the output of a Takagi-Sugeno FM is described. Because in Takagi-Sugeno FMs the *defuzzification step* is omitted[1] the following steps must be carried out:

$$\text{Fuzzification} \rightarrow \text{Aggregation} \rightarrow \text{Activation} \rightarrow \text{Accumulation}$$

The *fuzzification step* uses the MFs to map crisp model inputs to the degrees of membership. These degrees of membership will be denoted as $\mu_r^{A_n^{i_n}}(u_n)$, with r = rule index ($r \in 1,\ldots,R$, whereat R represents the number of rules in the rule-base), $A_n^{i_n}$ denoting the ith fuzzy set of input n ($n \in 1,\ldots,N$, whereat N is the number of model inputs) and u_n is the nth element of the model input vector $\underline{u}$.

The *aggregation step* combines the individual linguistic statements with the help of a fuzzy operation (see Sec. 3.2.2) to the degree of rule fulfillment. When the fuzzy model is in conjunctive form and the product operator is used as t-norm the degree of fulfillment of rule r is:

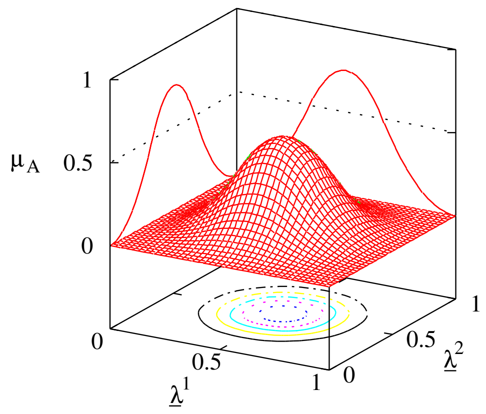
$$\mu_r(\underline{u}) = \mu_r^{A_1^{i_1}}(u_1) \cdot \mu_r^{A_2^{i_2}}(u_2) \cdot \ldots \cdot \mu_r^{A_N^{i_N}}(u_N). \tag{3.8}$$

Figure 3.4 illustrates two bivariate MFs, constructed by multiplying two univariate b-splines. In the artificial neural network community the support of the resulting n-variate function is often denoted as *receptive field*.

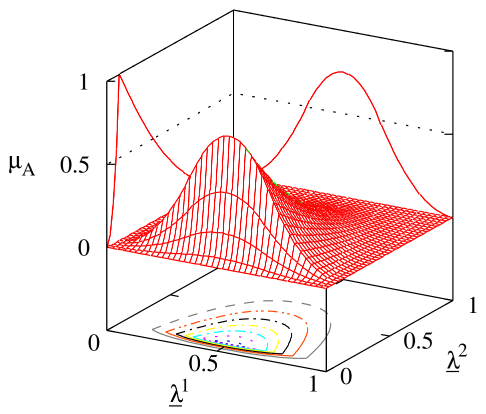
The *activation step* is used to calculate the output activations of the rules. Linguistic fuzzy models, for example, often uses the `min`-operation to cut the output MFs at the smallest degree of rule fulfillment. In the case of Takagi-Sugeno FMs the activation step comprises the task of calculating a function whose output is taken as degree of rule fulfillment.

The *accumulation step* is used to join all rule activation values together. In the case of linguistic FMs the accumulation step yield in a fuzzy set, by applying, for

[1] To be consistently with Fig. 3.2 the defuzzification step in Takagi-Sugeno FMs can also be seen as a function which maps each value to itself.



(a) Aligned bivariate b-spline.



(b) Displaced bivariate b-spline.

Figure 3.4: Bivariate b-splines formed by univariate b-splines. The aligned bivariate b-spline (a) is formed by two univariate b-splines of order three with knot-vectors $\underline{\lambda}^1 = \underline{\lambda}^2 = (0, 0.\overline{3}, 0.\overline{6}, 1)$. The displaced bivariate b-spline (b) is formed by two univariate b-splines of order three with knot-vectors $\underline{\lambda}^1 = (0, 0.\overline{3}, 0.\overline{6}, 1)$ and $\underline{\lambda}^2 = (0, 0.1, 0.1001, 1)$.

example, the `max`-operation. In the case of Takagi-Sugeno FMs the accumulation step yields in the final model output, which is normally calculated as a weighted average of all rule activations.

3.3 Interpretability Conditions of Fuzzy Models

Each fuzzy-modeling process has to deal with an important trade-off between interpretability and accuracy of the model. By fulfilling all criteria which support a good interpretability the FM is heavily restricted and logically the accuracy is inferior as a FM disregarding all interpretability consideration. In fact the author has the opinion that many kernel based models[2] which are termed as ”(interpretable) FMs” are de facto not interpretable. This is caused by the fact that many authors disregard the semantic consistency of the resulting model. The most basic assumption is that a fuzzy

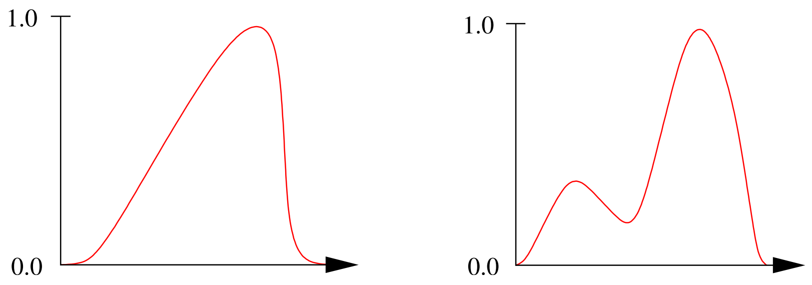
[2]Nearly all kernel based methods can be seen as FMs. For example local linear models or radial basis function networks were shown to be equivalent to zero order TSFMs [93, 132, 113]. Also support vector machines or wavelet networks are strongly related to fuzzy models because the used kernels can be interpreted as fuzzy sets, drawing the kernel based method into a fuzzy framework.

set has to be convex. Convexity of fuzzy sets can be defined in terms of its α-cuts, where an α-cut of a fuzzy set A is a crisp set A_α that contains all the elements of the universal set U that have a membership grade in A greater than or equal to α. A fuzzy set is convex (see also Fig. 3.5(a)) if and only if each of its α-cuts is a convex set. Or, equivalently, a fuzzy set A is convex if and only if

$$\mu_A(\lambda u_1 + (1-\lambda)u_2) \geq \min(\mu_a(u_1), \mu_A(u_2)), \qquad (3.9)$$

$$\text{with} \quad u_1, u_2 \in U, \lambda \in [0,1].$$

Note that this definition of convex fuzzy sets does not imply that the membership function of a convex fuzzy set is a convex function.



(a) Convex fuzzy set (b) Non-convex fuzzy set.

Figure 3.5: Convex and non-convex fuzzy set.

Kernel based approaches which do not fulfill the convexity condition (see Fig. 3.5(b)) can, by definition, not be termed as FMs. For an extensive and more mathematically based inspection of fuzzy set semantics the interested reader is referred to [107]. Some remarks about convexity and complementary or non-complementary MFs with respect to the output of FMs can be found in [66].

Other conditions are not so fundamental and kernel based approaches violating these conditions can, by definition, be denoted as FMs. Besides the indispensable convexity of fuzzy sets, the interpretability of FMs depends on the fulfillment of other conditions.

3.3.1 Fuzzy Set Configurations Causing Semantic Inconsistency

To clarify the term sematic inconsistency some undesirable fuzzy set configurations are shown. The following figures show a linguistic variable called ”annual salary” covered by two fuzzy sets which are labeled POOR and WEALTHY.

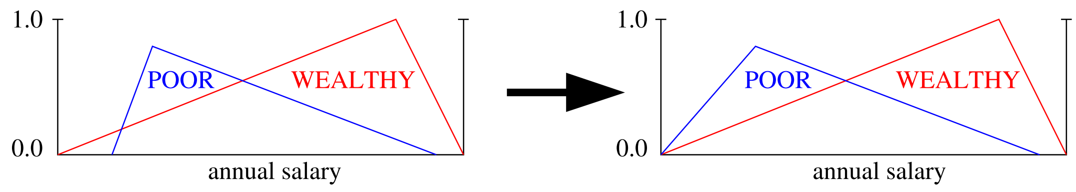


Figure 3.6: Fulfilling the ordinal condition of fuzzy sets.

Figure 3.6 illustrates the ordinal condition. The left hand side fuzzy set distribution has as semantic consequence that a person could be WEALTHY to a certain degree, but the same person is simultaneously not even POOR. Obviously a proper semantic interpretability is not given. The right hand side of Fig. 3.6 illustrates the corrected version of the left hand side, now fulfilling the ordinal condition.

Figure 3.7 illustrates the leveling condition (i.e. the special case of normalization to one). The left hand side of Fig. 3.7 exhibits different degrees of maximal fulfillments for two fuzzy sets covering the same linguistic variable (i.e. annual salary). The semantical consequence is that a POOR person is never as POOR as a WEALTHY person is WEALTHY. Applying double standards is in general not desirable. Again, the right hand side of Fig. 3.7 provides, with respect to the leveling condition, a correct version of the left hand side.

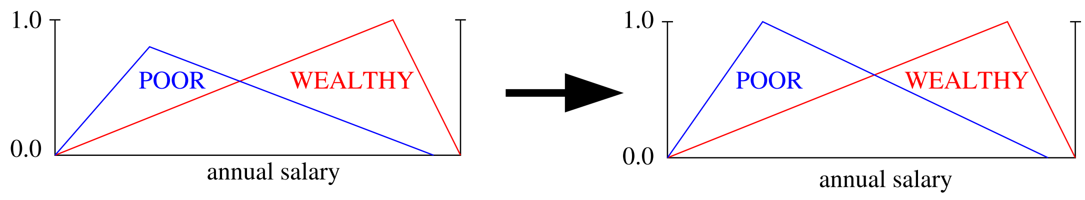


Figure 3.7: Fulfilling the leveling (i.e. normalization to one) condition of fuzzy sets.

Figure 3.8 illustrates the *complementarity condition*. A person classified as 100% POOR (kernel activation equals one) can not be partly WEALTHY or vice versa. It is always desirable that the activations of all inputs sum up to an equal value. This property is denoted as *partition of unity* or as complementarity condition.

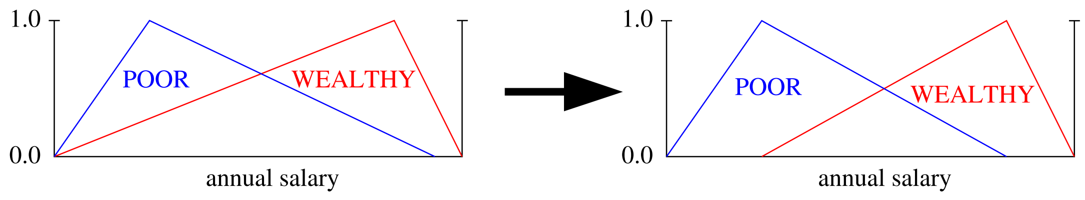


Figure 3.8: Fulfilling the complementarity condition of fuzzy sets.

Figure 3.9 again illustrates the complementarity condition. The attentive reader will have realized that the fuzzy sets (see right hand side of Fig. 3.8) does not everywhere fulfill the complementarity condition. In fact the right hand side of Fig. 3.9 illustrates the one and only possible fuzzy set distribution for two triangular (not trapezoidal) fuzzy sets fulfilling all interpretability conditions for a semantically correct interpretation. Corresponding to the above examples b-splines of order two would yield in semantically correct MFs.

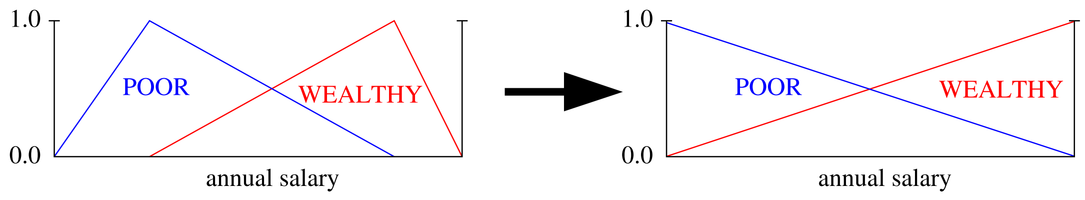


Figure 3.9: Fulfilling the complementarity condition of fuzzy sets part two.

3.3.2 Interpretability Factors

Beside these sematic conditions there exist other factors which influence the interpretability of a fuzzy model. Unfortunately an objective and commonly used measure for model interpretability is still not available. Thus, reasonable accuracy comparisons of FMs are restricted to models using the same model structure. The interpretability factors can be classified into two different classes. The first category is concerned with the fuzzy set distribution of a FM as exemplarily shown in Sec. 3.3.1 and the second category is concerned to the rule-structure of a FM.

The fuzzy set distribution, sometimes denoted as *semantic criteria*, is very important, since every fuzzy set should represent a linguistic term with a clear semantic meaning. The focus of fuzzy set distribution should lie on the meaning of the ensemble of labels instead of the absolute meaning of each term in isolation. Thus, the following criteria should be fulfilled by a fuzzy set distribution to facilitate the task of assigning linguistic terms as good as possible.

Complementarity: For each input element of the universe of discourse, all membership values of the fuzzy sets should sum up to one. This characteristic is known as forming a *partition of unity* and guarantees a uniform distribution of meaning among the elements. Furthermore a fuzzy set distribution forming a partition of unity improves the learning capabilities [187, 66] of a FM.

Leveling: The leveling condition is a generalized form of the most of the time unnecessary strict, but commonly known, normalization (to one) condition, which claims that at least one input element of the universe of discourse should activate the membership value equal to one. A normalization to one has as consequence that there exist an absolute truth, e.g. an age were you are only old, a state were a liquid is only hot and so on. This is often contradictory to the reasons why fuzzy concepts were introduced. The more general leveling condition fits better in a fuzzy framework because it claims only an equalized maximum activation level of all fuzzy sets of the same linguistic variable. For the sake of simplicity Fig.3.6-Fig. 3.9 only made use of normalized triangular MFS. However, by using b-splines of higher order ($k > 2$) the leveling condition could be implemented. In the case of b-splines, which are defined over uniformly distributed knots, the leveling condition implicitly holds (Fig. 3.3(c)).

Human manageable number of elements: Since a human being can handle only a certain number of conceptual entities ($\approx 7 \pm 2$), the number of MFs covering one input should not exceed this quantity.

Distinguishability: Each linguistic label should have semantic meaning and the fuzzy set should clearly define a range in the universe of discourse of the input variable.

Beside the fuzzy set distribution the rule structure has to be taken into account to define an overall interpretability measure. As discussed in section 3.2.2 a fuzzy rule relates one or more input-variable conditions via a t-norm, called antecedents, to a corresponding output conclusion, called consequent. The interpretability of a FM is influenced by the structure of each rule and by the whole set of rules. These rule-structure criteria, also called *syntactic criteria*, for interpretability are given by the following conditions.

Completeness: A fuzzy rule based model should infer for each conceivable model input a corresponding output. In addition to this, some authors claim [9, 37] that the fuzzy set obtained by combining all the individual rule outputs, has to be non empty. This requirement is unnecessary strict, since the main objective of the completeness condition is to assure that the model output is always well defined. In the following this objective will be fulfilled by simply introducing a default model output of zero. This results in an always well defined and smooth[3] output behavior of the model.

Consistency: The consequents of two or more simultaneously firing rules with the same antecedents should not be contradictory but semantically close.

Readability: The number of conditions (premises) of each rule should not exceed the human manageable number of conceptual entities $(\approx 7 \pm 2)$.

Simplicity: In general it is desirable to model a system by an FM with only a few rules, but the overall number of rules has no direct influence on the interpretability properties of a FM. Note that there exist very complex real world sequences which can be described by highly specialized human experts. If this description would be mapped into a mathematical model, the number of rules would be enormous, but the interpretability would still be given. The simplicity condition should be seen as the demand that as few as possible rules are activated simultaneously in order to enable a local view of the behavior.

3.3.3 An Exemplary Interpretability Measure

With an objective interpretability measure (IM) it becomes possible to compare the accuracy of FMs using different structural assumptions. All the above mentioned in-

[3]The output will be denoted as smooth if all rules use convex and non-crisp fuzzy sets. For example the usage of b-splines (no coincident knots) of order three or higher as fuzzy sets always produce a smooth model output without discontinuities.

terpretability conditions can be subsumed in only five objective interpretability factors (IF). These five IFs, each with values in the range [0,1], can than be joined by a product operation to form an objective IM, which again is always in the range from zero (no interpretability) to one (best interpretability):

$$IM = \prod_{i=1}^{5} IF_i \tag{3.10}$$

with interpretability factor IF_1 concerning the number of used premises:

$$IF_1 = \frac{R}{\sum_{r=1}^{R}(2 - \mathtt{max}(3, NoP_r))^2} \tag{3.11}$$

R	:	number of rules,
NoP_r	:	number of premises of rule r.

The numbers two and three in Eq. (3.11) are used to control the maximum number of used premises in one rule. In this case one up to three premises lead to an interpretability factor of one (full interpretable). By using more premises the interpretability decreases quadratically. The chosen numbers and the quadratic factor reflect the observation that human beings can hardly understand rules with more than three premises.

Interpretability factor IF_2 concerns the number of fuzzy sets covering the inputs:

$$IF_2 = \frac{N}{\sum_{n=1}^{N}(6 - \mathtt{max}(7, NoFS_n))^2} \tag{3.12}$$

N	:	number of model inputs,
$NoFS_n$	:	number of fuzzy sets covering input n.

Here the numbers six and seven reflect the observation that human beings can hardly manage more than seven $(\approx 7 \pm 2)$ conceptual entities. Again, if more conceptual entities (fuzzy sets) are used, the interpretability factor decreases quadratically.

Interpretability factor IF_3 concerns the number of simultaneously activated rules:

$$IF_3 = \frac{M}{\sum_{m=1}^{M}(3 - \mathtt{max}(4, NoAR_m))^2} \tag{3.13}$$

M	:	number of data patterns,
$NoAR_m$	:	number of activated rules.

In Eq. (3.13) the numbers three and four are chosen due to a compromise concerning the values used in Eq. (3.11) and the fact that already a problem with two inputs, each input covered by fuzzy sets of order two, leads to four simultaneously activated rules (in the case of a complete rule-base using no ”dont’t care” rules). Again, a quadratic decrease in interpretability is implemented by the quadratic term, which should reflect the capabilities of human beings.

Interpretability factor IF_4 concerns the different levels of the maximum activations of the fuzzy sets covering the same input:

$$IF_4 = \frac{N}{\sum_{n=1}^{N}(1 - \sqrt{MaxDiV_n})} \tag{3.14}$$

N	:	number of model inputs,
$MaxDiV_n$	:	maximal difference in activation values of fuzzy sets covering input n.

Obviously the square root operation in Eq. (3.15) should penalize ”higher” differences in the maximum activation of fuzzy sets more than ”lower” differences.

Interpretability factor IF_5 concerns the violation of the complementarity condition of fuzzy sets:

$$IF_5 = \frac{N}{\sum_{n=1}^{N} PV_n} \tag{3.15}$$

$$\begin{aligned}
N \quad &: \text{ number of model inputs,} \\
PV_n \quad &= \begin{cases} \frac{(u_n^{max}-u_n^{min})}{\sum_{j=1}^{k_n} \int FuzzySet_j} & \textbf{if} \quad \sum_{j=1}^{k_n} \int FuzzySet_j \geq 1.0, \\ 1+(1-\frac{(u_n^{max}-u_n^{min})}{\sum_{j=1}^{k_n} \int FuzzySet_j}) & \textbf{otherwise} \end{cases} \\
k_n \quad &: \text{ number of fuzzy sets covering input } n.
\end{aligned}$$

At best IF_5 has never to be calculated because the used fuzzy sets inherently supports the fulfillment of the complementarity condition (as it is the case by using b-splines as fuzzy sets). If for some reasons the integration is not possible, a ”big” number of uniformly distributed input patterns can be used to approximate IF_5.

3.3.4 Avoiding the Curse of Dimensionality

By using the local support areas of the fuzzy sets covering the inputs, it is possible to construct an n-dimensional lattice (n = number of model inputs). A rule-base in which each grid is considered by one rule[4] is denoted as a *fully defined rule-base.* Obviously the number of rules in a fully defined rule-base increases exponentially with the number of used inputs and thus the resulting models become infeasible for high-dimensional systems. Moreover, by considering a fully defined rule-base the number of premises for each rule is equal to the number of inputs, which violates for high-dimensional problems the above mentioned readability condition. To tackle these two problems, by preserving the interpretability, some authors use ”*don't care*” as a valid input label [92, 115]. Variables in a given rule that are labeled with ”don't care” are considered as irrelevant and thus, a rule like

IF eye-color IS ”don't care” AND hobby IS dangerous THEN injury risk IS high

is equivalent to the rule

[4]This means that each grid is covered by one n-dimensional function build by n one-dimensional MFs (e.g. b-splines).

IF hobby IS dangerous THEN injury risk IS high.

Another possible way to bypass the curse of dimensionality and to ensure interpretability is to use default rules with a firing strength inverse proportional to the firing strength of all other rules.

In the following, models with so called *partly defined rule-bases* will be used. Rules of a partly defined rule-base do not necessarily cover all input regions and furthermore they make use of ”don’t care” premises. By using a default model output of zero to assure an always well defined model output behavior (see completeness condition in Sec. 3.3.2) the resulting models are applicable to high-dimensional problems without loosing their interpretability.

3.4 Takagi-Sugeno Fuzzy Models

Takagi and Sugeno proposed in 1985 a FM with a rule-base comprising R rules of the form:

$$\text{Rule}_r : \text{IF } u_1 \text{ IS } A_1^{i_1} \text{ AND } \cdots \text{ AND } u_N \text{ IS } A_N^{i_N} \text{ THEN } \hat{y}_r = f_r(\underline{u}), \tag{3.16}$$

$$\begin{aligned} \text{with} \quad A_n &: \text{ fuzzy set of the } n^{\text{th}} \text{ linguistic variable,} \\ i_n &: \text{ fuzzy set number of the } n^{\text{th}} \text{ linguistic variable,} \\ N &: \text{ length of input vector } \underline{u}, \\ \hat{y}_r &: \text{ output of rule } r. \end{aligned}$$

Since the functions $f_r(\cdot)$ are not fuzzy sets, in most cases Takagi-Sugeno fuzzy models (TSFM) are hard to interpret. Note, that there are two exemptions to this general statement. The first exemption is that TSFMs possess excellent interpretation for dynamic processes [137]. The second exemption can be established by choosing $f_r(\cdot)$ to be a real value s_r. The value s_r is denoted as *singleton* of rule r and is a special case of a fuzzy set. Thus, the interpretability of a singleton TSFMs is maintained. TSFMs utilizing constant functions (singleton FMs) as rule-consequents are also denoted as zeroth order TSFMs, TSFM with linear functions as rule-consequents are called *first order TSFMs,* etc.

However, each rule output is a function of the input vector and the overall model output $\hat{y}$ of a Takagi-Sugeno fuzzy model (TSFM) is mostly calculated as a weighted sum of all rule outputs $\hat{y}_r$:

$$\hat{y} = \frac{\sum_{r=1}^{R} f_r(\underline{u})\mu_r(\underline{u})}{\sum_{r=1}^{R} \mu_r(\underline{u})}, \tag{3.17}$$

with R : total number of rules,

μ_r : aggregated premise activation of rule r (e.g. Eq. 3.8).

The aggregated premise activations are also denoted as the rule firing levels, which are defined as the output of Eq. (3.7) utilizing fuzzy sets of rule r

$$\mu_r(\underline{u}) = h(\mu_r^{A_n^i}(u_n)), \tag{3.18}$$

where h is an arbitrary t-norm. In the case of TSFMs h usually denotes the algebraic product or minimum operation. An unnormalized version of Eq. (3.17), as proposed by [179, 180], has for its output

$$\hat{y} = \sum_{r=1}^{R} w_r \mu_r(\underline{u}), \tag{3.19}$$

with w_r : real value s_r (singleton) of rule r.

In the following the algebraic product will be used as t-norm. Obviously the denominator in Eq. (3.17) forces a normalization, which is, in general, necessary to assure the interpretability of the FM. However, if the rule-base is fully defined and, as assumed, the algebraic product is used as t-norm and furthermore the partition of unity holds the denominator can be canceled. The resulting kind of TSFM is equivalent to a normalized lattice based (radial)-basis function network [93, 98]. Furthermore, the extension of this result, namely that TSFMs are equivalent to local model networks, was shown by [77].

3.4.1 Takagi-Sugeno Fuzzy Models for System Identification

Zero$^{\text{th}}$ order TSFMs are a widely applied model class in industry to tackle static problems because of interpretability issues and because the rule-consequences can be estimated simultaneously in a single LS optimization. This single step estimation is referred to as *global estimation*. Unfortunately many users totally negate the interpretability issues by freeing all constraints which could yield in an interpretable fuzzy set distribution. By doing so one of the main advantages of zero$^{\text{th}}$ order TSFMs vanishes and the user would be better off using TSFM of higher order or other approximation techniques.

3.4.1.1 Approximate Takagi-Sugeno Fuzzy Models

Approximate TSFMs(ATSFMs) have an incomplete rule-base and the interpretability is generally lessened or nonexistent since there exists no global term set definition. Each rule defines its own MFs on arbitrarily inputs, generally violating the interpretability conditions stated in section 3.3.2. Consequently ATSFMs do not suffer from the curse of dimensionality and thus, it is easy to construct an ATSFM with fewer rules but the same prediction quality as it is possible with a higher constraint but more interpretable TSFM.

3.4.1.2 Descriptive Takagi-Sugeno Fuzzy Models

The main criterion to denote a TSFM as ”descriptive” is the existence of a global term set definition for each input variable [25]. Unfortunately descriptive TSFMs (DTSFM) are not necessarily interpretable, since interpretability depends on more factors than a global term set definition. However, together with a high fulfillment of the in Sec.3.3.2 introduced interpretability factors, DTSFMs provides the most interpretable class of TSFMs; capable to use the powerful and well established methods of single step LS to determine the model parameters.

3.4.2 Parameter Estimation of Takagi-Sugeno Fuzzy Models

For the remainder the term ”parameter optimization” always refers to optimization of the consequent parameters. The consequent parameters correspond to the output weights in a basis function formulation or the so called coefficients in a spline approximation formulation. These parameters can easily be calculated by single step LS methods as described in Sec. 2.4.2 and [38, 132, 157]. This simplicity of optimal parameter estimation is a main reason to utilize TSFM in data-driven modeling. Of course parameter optimization of TSFMs can also be performed by iterative gradient methods [18]. These methods are preferable if fast to calculate low-accuracy solution are needed, for example in algorithms which operate in real-time. Another reason to use iterative gradient methods is the computational cost[5] and the large memory requirements of direct methods. If non of these reasons are important, direct methods are preferable and thus, in the following only direct optimization techniques are considered.

3.4.3 Global Parameter Estimation of Takagi-Sugeno FMs

By using global estimation all parameters of a model are estimated in a single LS optimization. If we assume complete polynomials as rule consequents the parameter vector (in the consequent proposition) of a one rule of a TSFM contains

$$P = \frac{(N+k)!}{N!k!}, \tag{3.20}$$

$$\text{with} \quad N: \text{ number of model inputs,}$$
$$k: \text{ order of the TSFM,}$$

parameters (see also Eq. 2.20). Thus, the total number of parameters which have to be estimated in a TSFM is

$$W = R \cdot P, \tag{3.21}$$

$$\text{with} \quad R: \text{ number of rules,}$$
$$P: \text{ number of rule parameters.}$$

[5]Direct optimization techniques are generally based on performing a matrix inversion that has a computationally cost of $O(p^3)$.

In the following the parameter vector containing all parameters of a TSFM is denoted as weight vector and is for zero order TSFMs

$$\underline{w} = [w_1 \cdots w_R]^{\mathrm{T}}. \tag{3.22}$$

In the case of a first order TSFM the weight vector is

$$\underline{w} = [w_{1,1}\, w_{1,2} \cdots w_{1,P} \cdots w_{R,1}\, w_{R,2} \cdots w_{R,P}]^{\mathrm{T}}. \tag{3.23}$$

The associated regression matrix $\mathbf{X}$ for M available data samples becomes

$$\mathbf{X} = [\mathbf{X}_1^{\text{sub}}\, \mathbf{X}_2^{\text{sub}} \cdots \mathbf{X}_R^{\text{sub}}] \tag{3.24}$$

with the regression sub-matrices

$$\mathbf{X}_r^{\text{sub}} = \begin{bmatrix} \Phi_r(\underline{u}_1) & u_{1,1}\Phi_r(\underline{u}_1) & u_{1,2}\Phi_r(\underline{u}_1) & \cdots & u_{1,P}\Phi_r(\underline{u}_1) \\ \Phi_r(\underline{u}_2) & u_{2,1}\Phi_r(\underline{u}_2) & u_{2,2}\Phi_r(\underline{u}_2) & \cdots & u_{2,P}\Phi_r(\underline{u}_2) \\ \vdots & \vdots & \vdots & & \vdots \\ \Phi_r(\underline{u}_M) & u_{M,1}\Phi_r(\underline{u}_M) & u_{M,2}\Phi_r(\underline{u}_M) & \cdots & u_{M,P}\Phi_r(\underline{u}_M), \end{bmatrix}$$

where $\underline{u}_m$ denotes the m^{th} input vector of the available M data samples and $\Phi_r(\cdot)$ denotes the combined linguistic statements of the antecedents of rule r (e.g. an n-variate b-spline function). The model output is then given by

$$\underline{\hat{y}} = \mathbf{X}\underline{w}, \tag{3.25}$$

with $\underline{\hat{y}} = [\hat{y}_1\, \hat{y}_2 \cdots \hat{y}_M]^{\mathrm{T}}$. For zero and first order TSFMs the globally optimal weights can be calculated, as described in Sec. 2.4.2.1, in a single step. Because the global LS estimation is a very efficient way to optimize the rule consequents this method will be used in the following. However, global LS estimation is not always applicable since its computational complexity grows cubically with the number of weights. Concerning ”large” data sets (e.g datasets with more than ≈ 10000 samples) local estimation methods has to be used, which only grow linearly with the number of parameters [136].

3.5 Structure Identification of Takagi-Sugeno FMs

From a system identification point of view, structure identification of a TSFM matches the task to select a model structure (see Sec. 2.3). In the case of TSFMs, structure selection subsumes the task to:

- Select a pool of independent variables which are potentially related to the desired output y.
- Cover these selected variables (features) with fuzzy sets.
- Select the shape of the fuzzy sets.

Obviously all three tasks should be done in an optimal way to achieve a good accuracy in combination with ideal generalization abilities.

3.5.1 Feature Selection

Feature selection methods become necessary if the dependencies of available information to a system output of interest is unknown. Selecting a set of features which is optimal for a given task is a problem which plays an important role in a wide variety of contexts and applications. This includes pattern recognition, adaptive control, machine learning, data-mining and modeling.

Methods for feature selection reaches from statistical tools and the classical greedy-algorithm [28], over graphical analysis [14] to global search methods like evolutionary computation. The first two approaches yield normally in suboptimal feature selections and the graphical analysis is hardly applicable in purely data-driven approaches. Evolutionary search methods provide, at least theoretically, an easy to implement global search method for optimal feature selection.

3.5.2 Input Space Partitioning

Input space partitioning provides information for fuzzy set positioning. There exist three common methods to partition the input space:

1. Clustering methods [3, 67, 75, 69, 76] like C-means.
2. Decision tree based methods like CART [15] or LOLIMOT [132].

3. Global search method like evolutionary computation [25].

By using clustering methods, the identified clusters are projected[6] on the input axes. The space between the cluster borders are used as the support of fuzzy sets. Soft clustering methods also provide utilizable information about the fuzzy set shape. Decision tree based approaches split the input space. This is done recursively in input areas where a selected model produces the highest classification/approximation error. The splitting is normally done axis orthogonal and thus directly usable as information to place fuzzy sets. The third method is the most general. Especially evolutionary computation has as advantage that, at least theoretically, it is possible to identify an optimal input space partition for fuzzy sets.

3.5.3 Fuzzy Set Shape Selection

The selection of an optimal[7] shape for each fuzzy set is almost always a highly nonlinear problem. Again, the usage of classical mathematical methods [95] to identify the (absolute) optimal shape are applicative only for very small problems. But to preserve the possibility of an (absolute) optimal fuzzy set shape selection a global search method has to be chosen.

3.6 Summary

In this chapter the structure of TSFMs was presented. The main focus pertained the interpretability of FMs. It was clarified that a descriptive TSFM is not necessarily interpretable. A usable interpretability measure based on five interpretability factors was introduced. Furthermore, the generally necessary tasks to achieve optimal TSFMs were listed and possible strategies to fulfill these tasks were presented. It was justified that evolutionary computation is an appropriate concept for structure identification of data-driven TSFM. Because of this, the next chapter gives a problem specific introduction to evolutionary computation. These concepts, in combination with a novel method to formulate the genotype search space of candidate solutions with the help of grammar-based genotype templates, will later be used to identify optimal DTSFMs.

[6]The clusters can also be directly interpreted as multi-dimensional fuzzy sets. This leads to approximate TSFMs.

[7]Optimal with respect to model accuracy and model interpretability.

Chapter 4

Evolutionary Computation

The idea of *evolutionary computation* (EC) is to utilize principles of evolution in nature to find (sub)optimal solutions to NP-complete problems. Thus, most ideas used in EC have their origin in observations of biological based functionality. The basic functionality of natural evolution was first correctly described and published by Charles Robert Darwin (see Fig. 4.1) in his famous book on ”The Origin of Species” [30] from 1859.

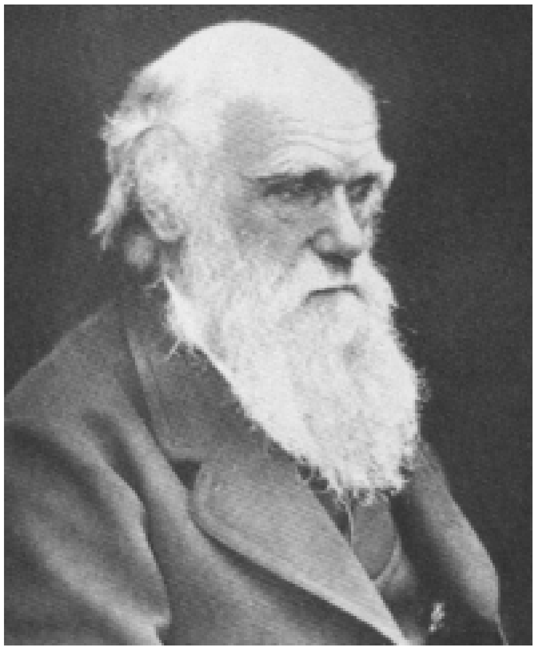


Figure 4.1: Charles Robert Darwin (⋆ 12th February 1809, † 19th April 1882).

Thus, artificial evolution considers a solution to a given problem as an individual and a set of solutions as a population, the term EC subsumes all population based approaches which possess the following characteristics:

- Random variation of individual solutions.
- Alteration of potentially useful structures to generate new solutions.
- A selection mechanism to increase the proportion of better solutions.

The schematic functionality of EC is depicted in Fig. 4.2 and the classification with respect to other search methods [52] is illustrated in Fig. 4.3. In the following the indi-

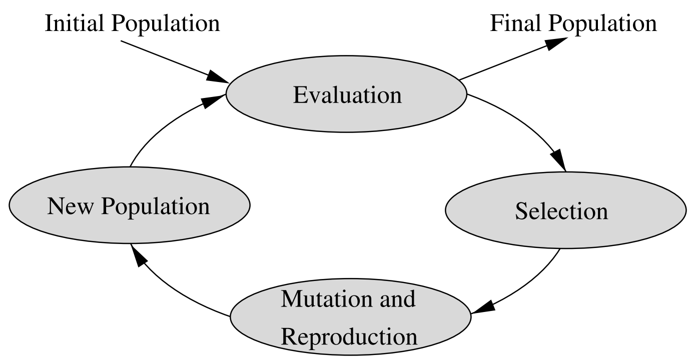


Figure 4.2: Schematic functionality of evolutionary processes.

viduals of a population are denoted as $Indi_i (1 \leq i \leq I)$ where I stands for the number of individuals in the according population. Furthermore a population, although not sorted, will be represented by a tuple $Pop_t = \langle Indi_i \rangle_{1 \leq i \leq I}$, with t as iteration (generation) index. This notation is caused by the fact that several individuals in the same population can be identical, thus, a set-theory based nomenclature would be cumbersome.

At a first glance EC seems not to introduce new concepts with respect to more traditional search methods like, parallel simulated annealing [150] or parallel tabu search [71]. The main difference of population based algorithms to all other techniques is the concept of competition between candidate solutions.

Different characteristics of EC leads to special evolutionary algorithms (EAs), with names like genetic algorithms [73] (GAs) , evolutionary strategies [154, 164] (ESs) , evolutionary programming [53], genetic programming [104] or artificial immune systems [33]. To shortly outline the functionality of an EA or to do research

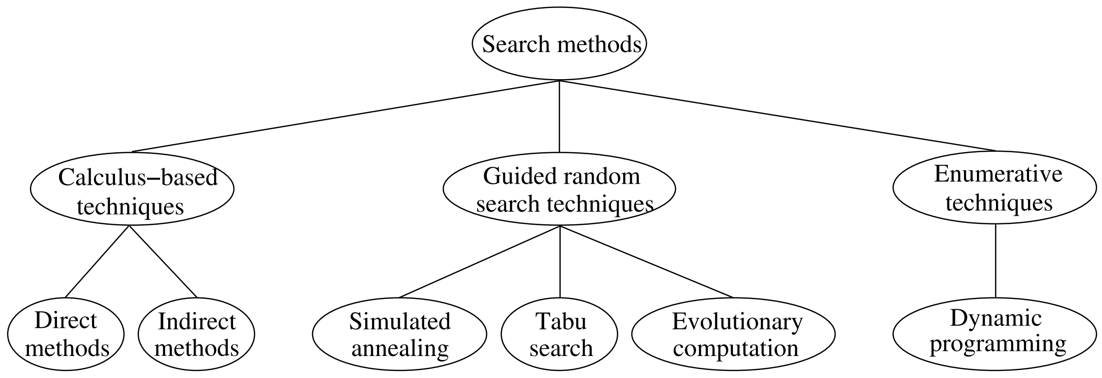


Figure 4.3: A taxonomy of search methods.

in theoretical analysis of fundamental qualities, these terms are very useful. From the perspective of most users of EC concepts it is unimportant which strategy is used, beside the fact that the found solution is satisfactory. For this reason nowadays most EC approaches use a set of characteristics depending on the problem to solve, yielding hybridization of several methods. This has as consequence a decoupling from classical terminology with its strict categorization, leading to an explanation of used EC concepts.

4.1 Nomenclature of Evolutionary Computation

As already mentioned EC utilizes principles of natural evolution. Because of this and for simplicity EC make use of terms with their origin in biology:

- Individual - a particular biological organism.
- Fitness - measurement that express the success of an individual to handle its living conditions.
- Population - group of interbreeding individuals within a given area.
- Phenotype - refers to the composition of a particular individual.
- Gene - is a functional entity that encodes a specific feature of an individual.
- Genotype/DNA - refers to a specific combination of genes carried by an individual and can be seen as a blueprint which is transcribed into proteins which build the phenotype.

Keeping the limitations of nomenclature transfer between different scientific areas in mind, the above-mentioned terms have in EC the same meaning with the exception of the following modifications:

- Individual - particular solution to a certain problem. An individual in EC subsumes the genotype and the phenotype.
- Fitness - measurement computed via a fitness function which expresses the success of an individual to solve the problem.
- Genotype[1] - blueprint which can be transcribed via function(s) into a phenotype.

4.2 Genotype Representation

Evolution, like all search algorithms, is limited and constrained by the representation (i.e. genotype) it can modify. A *genotype representation* is a mapping from the state space of possible encodings to the state space of a genotype. To fully understand this statement it is necessary to realize that the location of information carried by a genotype is twofold. Firstly, and most apparent, information is embedded in the structure of the genotype. In the case of natural genotypes this is the base pair sequence and for a string based artificial counterpart it is each element (Bit) of the considered string. Secondly, and not so obvious, the structure for itself can carry information. This structure is meant by genotype representation. Already Koza [103] mentioned that:

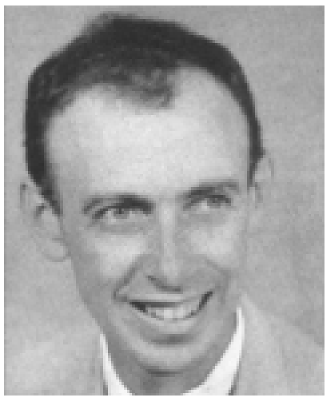
> ”Representation is a key issue in genetic algorithm work because the representation scheme can severely limit the window by which the system observes its world.” [. . .] ”String-based representation schemes are difficult and unnatural for many problems and the need for more powerful representations has been recognized for some time [34, 35, 36].”

Therefore, and to familiarize the used terminology, the following section starts with a brief discussion of the biological archetype of a genotype representation and their algorithmic counterparts.

[1]In EC literature the terms genotype, chromosome and DNA (seldom used) are mostly used interchangeably. In the following the term genotype is used to describe a set of genetic parameters that encode a candidate solutions, because this is closer to the biological meaning.

4.2.1 The Biological Genotype

The *genotype* (blueprint) of each living entity on this planet is encoded in the *deoxyribose nucleic acid* (DNA) . The structure of DNA was discovered by James Watson and Francis Crick [89] (see Fig. 4.4) in 1953.


(a) James Watson

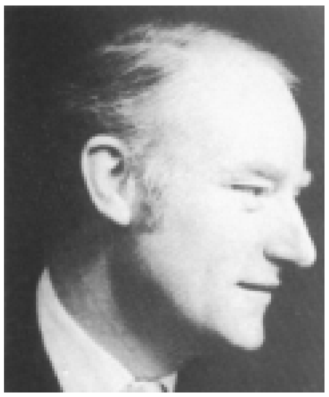

(b) Francis Crick

Figure 4.4: Watson and Crick, the discoverer of the structure of DNA.

DNA is shaped like a twisted step-ladder also known as a double helix (see Fig. 4.5). The genetic information is carried on the rungs of the ladder. In reality each rung is equivalent to a base pair formed by two nucleotide bases and the supporting bannister is formed by sugar and phosphate. Four nucleotide bases (adenine, thymine, guanine and cytosine) are used in the DNA and only four different combinations of base pairs are possible, because adenine always goes with thymine and guanine always goes with cytosine. The creation of an organism, also called *phenotype*, from this genotype is a complex process. Because the details of this process are not within the scope of this thesis, in the following only a short outline of the functionality is given. To value the potential to further research in EC, some in EA seldom implemented, biological functionalities are mentioned.

Consistently three base pairs (called a codon) encode a certain amino acid. Since there exist $4^3 = 64$ possible combinations to form a codon and only 22 natural amino acids[2], the genetic code is redundant. Each DNA segment containing the information

[2]As things are now there are 22 known natural amino acids, possibly there are more.

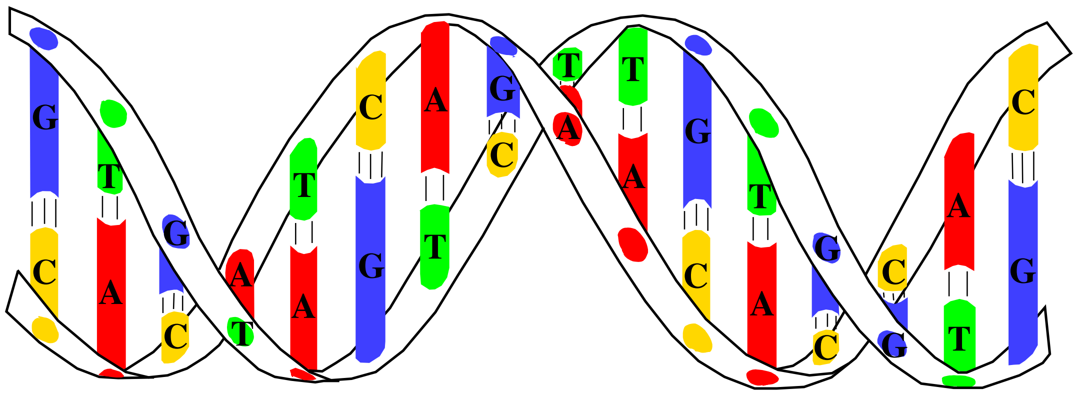


Figure 4.5: Schematic structure of DNA with sugar-phosphate backbone and four different nucleotide bases, each base-pair joined by hydrogen bonds.

for making a protein constitutes a gene. The information in a protein-encoding gene is copied into a *messenger ribonucleic acid* (mRNA) molecule that moves to so called ribosomes. A *ribosome* moves along a mRNA molecule, reading the codon for protein assembly as it goes. As it moves, the ribosome assembles amino acids into a gradually lengthening protein chain. At the end of the coded message, translation stops, the ribosomal subunits separate and detach from the mRNA, and the completed protein is released. While DNA stores the information, *proteins* are the actors in each cell. The functional tasks proteins perform are various. Proteins give form and elasticity to tissue, they transport and store material and information, they recognize and bind foreign substances and signal proteins, they catalyze biochemical processes and they stimulate cell division. Thus, proteins form the machinery which build up a phenotype. Research estimations numeralize the number of different proteins in the human body to 100000. It is important to state that information is only passed from genotype to phenotype.

$$\text{Biological Genotype (DNA)} \longrightarrow \text{Proteins} \longrightarrow \text{Phenotype}$$

The impact of genes on phenotype features of an organism can seldom be described by a simple one-to-one correspondence between genes and features. Several genes can have an influence on one phenotypical characteristic, which is referred as *polygeny*. On the other hand the term *pleiotropy* describes the effect that a single gene affects many phenotype features.

In the following we should always have in mind that this description of the biological process is by far not complete nor that the process is fully understood. For example, positional factors in the ovum are suspected to have influence on the gene expressions and some proteins for them self influence gene expressions too. Furthermore there exist theories that some proteins called *prions* (proteinaceous infectious particles) inherit specific protein attributes absolutely decoupled from information provided by the DNA.

4.2.2 Non-Coding Genotype Segments

Non-coding segments, also referred as *introns* in biological and EC literature, are genes which are not used in the genotype to phenotype mapping process. In contrast to non-coding segments all coding segments are subsumed by the term *exon*. In biology the term intron describes in fact only one kind of non-coding DNA which can be found within, but not between, genes. Another familiar type of non-coding segment in biology is defined by *promoter*/*terminator* sequences, referred as ptGA in EC literature [121], which defines start and end points of a coding sequence, respectively. Because in general this distinction is not made in EC literature, in the following the term non-coding segment is used.

Human DNA consists of approximately 97% non-coding DNA and only 3% coding DNA. The maintenance of such a large amount of non-coding DNA and the therewith aligned bit of extra processing for the biological organism, leads to the assumption that there must be an advantage to having it in the genome. Because non-coding segments are disregarded in the genotype to phenotype mapping process, they obviously do not contribute to the overall fitness of the individuals. Intuitive motivations for their existence in biological systems are that non-coding segments may guard against the disruptive effects of crossover, promote diversity, provide natural backups for the coding regions, and thus possibly expedite the evolution of better adapted individuals. It is also hypothesized that non-coding segments of genotypes enable a variable combination of coding segments they separate, a process called *exon shuffling* [40,58,59]. Thus, the introduction of a counterpart concept in EC could have the same effects on candidate solutions to a problem.

4.2.3 Artificial Genotypes

As mentioned above, a genotype is defined and consists of a structure (the double helix) and the information embedded in this structure (the base pair sequence). By using digital computers, a straightforward way to implement an artificial genotype is to use any data-structure which can carry information and to use a function to map this genotype to a phenotype (Artificial Genotype $\rightarrow$ Function(s) $\rightarrow$ Phenotype).

It is important to say that in artificial genotypes this information is not compelled to be used for the genotype to phenotype mapping, but it can also be used to parameterize the operators (for example mutation) working on the genotype representation.

It is comprehensible that in the early days of EC the genotype was chosen as simple as possible, firstly to simplify the implementation and secondly to analyze the fundamental behavior of operations on the genotype representation. Thus, the most common genotype was (and is) a structure consisting of memory cells stringed together. This memory cells could carry binary values as for example in GAs or real valued variables as for example in ESs.

One exception is the use of tree-structures for genotype representation in the field of genetic programming. This is founded in the fact that tree-structures can easily be parsed and mapped into a corresponding, for example LISP [1], programming expression. It is incomprehensible, that the tree-structure based genotype representation did not become popular in other evolutionary based methods, although a tree-structure implicitly supports the concept of building-blocks and hierarchical composition.

4.2.4 Fixed versus Variable Length Representation

For abbreviation artificial genotype representation is in the following sometimes referred as *encoding*. Natural evolution is open-ended with respect to the complexity of created life forms. EC applications that use only fixed length encoding do not share this advantage of open ended complexity. Thus, variable length encoding schemata have more expressive power and freedom to solve problems where the structure and size of a satisfactory solution is unknown in advance. Variable length representations in EC have been investigated since the early 1990's and the most famous representatives are the *messy GAs* [62] and the concept of genetic programming [105]. A good survey can be found in [70].

4.3 Solution Representation and Evaluation

Each evolutionary algorithmic approach for problem solving shows three basic components which are highly correlated and which have to be specified, namely:

The ***representation*** encodes alternative candidate solutions for manipulation.

The ***objective function*** describes the purpose to be fulfilled.

The ***fitness function*** returns a quality measurement for a particular solution.

The possible genotype representations of candidate solutions define the size and shape of the search space for the evolutionary algorithm. Because it is important to understand the consequences of different or inadequate representation and evaluation schemes the next sections provide a deeper insight into this problem. The provided information will later be used to justify the introduction of a novel and general representation scheme, which utilizes grammar based genotype templates to span a search space for evolutionary algorithms.

To evaluate the different candidate solutions it is necessary to distinguish between objective and fitness functions. By using the information provided by these evaluation functions an evolutionary algorithm tries to establish a gradient in the appropriate search space, which directs the subsequent individuals to ”better” search space regions. This and other essential considerations will be discussed in following sections.

4.3.1 Objective Function

An objective is something that an evolutionary optimizer seeks to accomplish or to obtain by means of his evolutionary operators. The evolutionary operators work on the genotype of a solution but the objective function judges about the characteristic of the phenotype. Thus, the objective function provides a gradient in the phenotype search space. Unfortunately, many problems have various objectives and the user has to decide which objectives have to be fulfilled. Often predefined values are used to weight the importance of each objective, but most time it is unknown in advance which objectives are mutually exclusive.

Difficulties can also arise if the ratio of feasible solutions is small and furthermore the objective function evolves no gradient to feasible phenotype regions. This could happen if all feasible solutions are equivalent in their objective measurement. For example the finding of a truth assignment that satisfies a well-formed Boolean expression (referred as satisfiability or SAT problem [57, 41]) leads only to objective values of $f(Indi) = 0$ or $f(Indi) = 1$, according to fulfillment or non-fulfillment of the expression. Similar difficulties arises in other combinatorial problems.

4.3.2 Fitness Function

The main difference to the objective function is that the goal of the fitness function (also called the *evaluation function*) is to provide a gradient for the evolutionary operators in the genotype search space. Nevertheless, the fitness has always to be proportional to the objective value, but the fitness function can yield several distinct fitness values for one objective measurement provided by the objective function. The fitness function, in the following denoted as $f(\cdot)$, returns a quality measurement for a particular solution, which can be used to compare different candidate solutions by the operator $\succ$. The operator $\succ$ is interpreted as ”is better than”, thus, for maximization problems $\succ$ denotes ”greater than” and for minimization problems $\succ$ denotes ”less than”. A possible approach for the above mentioned SAT problem is to calculate the fitness as the number of conjuncts that evaluate to true [39] or to change the Boolean variables into floating-point numbers [141] in the range $[0,1]$. It becomes clear that the fitness function is highly aligned with the genotype representation forming the genotype search space.

4.3.3 Search Space

The objective of nearly all real-world problems poses constraints, thus, setting up the objective functions includes the formulation of boundary conditions. These boundary conditions constitutes a *phenotype search space* (see Fig. 4.6). The phenotype search space is divided into a space of *feasible solutions* fulfilling the boundary conditions and a space of *infeasible solutions* which does not fulfill the constraints.

Logically the phenotype search space for feasible solutions is not necessary adjunctive nor convex and the ratio $\rho = \frac{|\mathcal{P}|}{|\mathcal{P}_{\text{feasible}}|}$ of feasible solutions can be approximately determined by randomly generating a huge number of random points from $\mathcal{P}$ and checking if they belong to $\mathcal{P}_{\text{feasible}}$ or not.

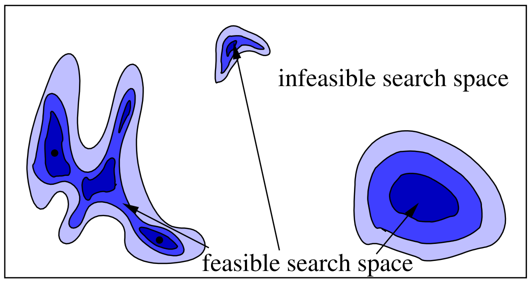


Figure 4.6: The rectangular area represents an exemplary two-dimensional phenotype search space $\mathcal{P}$, the blue areas the feasible and the white areas the infeasible parts.

4.3.3.1 Search Space Size

To point out the size of search spaces of even simple problems consider the task to find the highest output value $f(u_1,u_2)$ for the Schaffer function (see Fig. 4.7). The objective function, which in this case is identical to the fitness function, is to maximize:

$$f(u_1,u_2) = 0.5 - \frac{\sin^2(\sqrt{u_1^2+u_2^2}) - 0.5}{(1+0.001(u_1^2+u_2^2))}. \tag{4.1}$$

Lets assume we only need a solution of low accuracy (0.001 in a range of $[-10,10]$). For the sake of simplicity we encode both input variables u_1 and u_2 with 15 Bits, which leads to an input variable resolution of 0.00061. Thus, already this simply problem has a search space size of $2^{15} \cdot 2^{15} = 2^{30} = 1,073,741,824$ different states. Obviously the size of the search space is not determined by the objective function, but by the chosen representation.

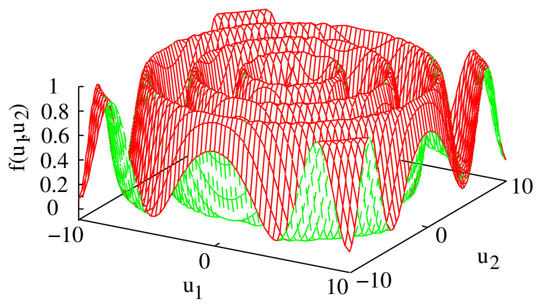


Figure 4.7: The Schaffer function.

4.3.3.2 Choice of an Appropriate Search Space

It is important so recognize, that by choosing the search space in an inappropriate or to restrictive manner, the possibility of numerous duplicate solutions or the precluding from any possible solution can occur. A descriptive example is given in [127], where the problem is to construct four equilateral triangles with six matches[3].

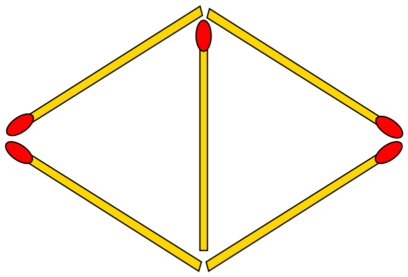


Figure 4.8: Two triangles, both fulfilling the equilateral constraint.

It is easy to construct two such triangles (see Fig. 4.8) or eight triangles only partly fulfilling the equilateral constraint (see Fig. 4.9). If we place the leftover sixth match in Fig. 4.9 from bottom left to upper right we will actually result in sixteen triangles, but again only the outer one is fulfilling the equilateral constraint.

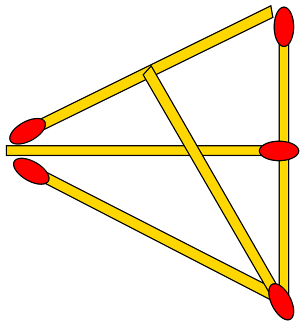


Figure 4.9: Eight triangles but only the outer triangle fulfills the equilateral constraint.

With the knowledge that the search space is incorrect, the interested reader may try to find the correct solution before turning over to the next page. To solve the task,

[3]In the original problem formulation the length of each triangle side should also be equal to the length of a match.

the search space has to be moved to three dimensions (see Fig. 4.10). By assuming a wrong search space, it is not possible to find the correct answer.

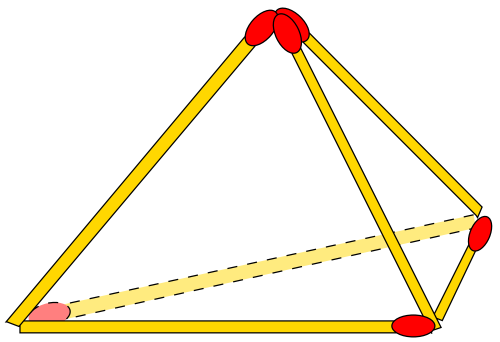
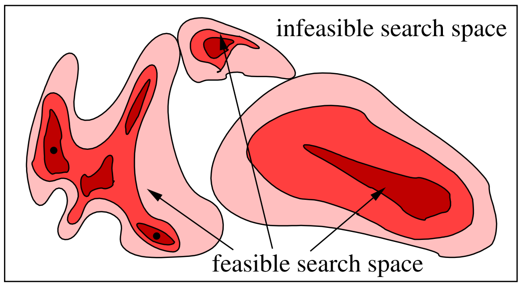


Figure 4.10: Four triangles, all fulfilling the equilateral constraint.

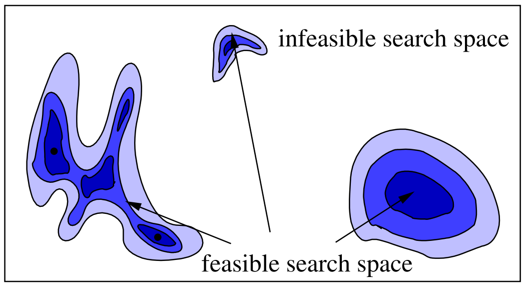
4.3.3.3 Genotype Search Space

The distinction made in Sec. 4.3.1 and Sec. 4.3.2 of objective and fitness functions was justified by the possible existing diversity of the phenotype search space and the *genotype search space*. It was stated that the evolutionary process follows a gradient in the genotype search space which is constituted by the fitness function. For the sake of simplicity the terms feasible solution and infeasible solution in the following are used twofold. Firstly by referring a phenotype solution and secondly to refer to a genotype representation which is mapped into a feasible/infeasible phenotype solution.

As mentioned above and again illustrated in Fig. 4.11(b) the search space $\mathcal{P}$ of feasible phenotypes is often not adjunctive. Thus, the primary goal of each genotype representation is to establish a genotype search space $\mathcal{G}$ such that the evolutionary search gets the possibility to follow fitness gradients in such a way that the distances between feasible regions of the search space and also the total size of the infeasible solution space is, in comparison to the phenotype search space, minimized. Often only this minimization of distances in the genotype search space makes it possible to bridge the infeasible solution gap between feasible solutions by single point mutations in the genotype. As consequence ”good” genotype search spaces should contain sets of genotypes connected by single point mutations that map into the same phenotype. This allows genetic changes to be made while maintaining the current phenotype and it also reduces the chance of becoming trapped in sub-optimal regions of the genotype search space. Using non-coding segments in the genotype is partly motivated (another reason is to memorize good and already used building blocks) by the same reason of supporting genetic changes based on single point mutations. Figure 4.11(a) illustrates exemplarily improved properties of the genotype search space in comparison to the corresponding phenotype search space depicted in Fig. 4.11(b).



(a) Possible genotype search space $\mathcal{G}$.



(b) Phenotype search space $\mathcal{P}$.

Figure 4.11: Possible modified genotype search space (a) and the concerning phenotype search space (b) given by the objective function.

For a comparison of different genotype to phenotype mappings see [169], where illustrative examples concerning the inter-phenotype accessibility are given. This concept of finding a ”good” redundant genotype to phenotype mappings is also followed by using decoders, a method which will be explained in the next section.

4.3.4 Infeasible Solution Handling

As discussed above, the genotype search space of possible solution encodings to a problem is normally divided into feasible and infeasible solution areas. Because only feasible solutions are of interest, it becomes necessary to deal with infeasible solutions. There exist three possible approaches in EC which can be used, namely avoiding infeasible solutions, repairing infeasible solutions or penalizing infeasible solutions. To obtain an understanding and to realize the pros and cons, each of these approaches is shortly outlined.

4.3.4.1 Avoiding Infeasible Solutions

One strategy to avoid infeasible solutions is based on preserving a feasible population of solutions by using special representations and evolutionary operators. By considering a disjoint integer valued string to encode candidate solutions for solving the traveling salesman problem, a special evolutionary operator would be the swapping of two integer values in the genotype. Several representations and specialized evolutionary operators were developed to tackle these kind of permutation problems.

Another interesting technique to avoid infeasible solutions is to use decoders [126, 106, 72, 64], where an arbitrary genotype search space $\mathcal{G}$ is mapped into the phenotype search space $\mathcal{P} = [-1, 1]^n$ by some decoders. Thus, the genotype does not encode the solution directly, but instead provides a set of instructions how to build a feasible solution. However, it should be noted that several factors should be taken into account while using decoders [140]:

- For each solution $p \in \mathcal{P}$ there is a decoded solution d.
- Each decoded solution d corresponds to a feasible solution p.
- All solutions in $\mathcal{P}$ should be represented by the same number of decodings d.
- Small changes in the decoded solution result in small changes in the solution itself.

Anyway, the more constrained a problem is, especially if constraints depend on other constraints, the more sophisticated it becomes to create appropriate operators or decoders. However, if it is possible to implement a strategy that avoids infeasible solutions, the arising advantages like no need for additional parameters and no need to evaluate infeasible solutions, is worth.

4.3.4.2 Repairing Infeasible Solutions

There exist two kinds of repair processes for infeasible solutions which have to be distinguished. Methods based on repairing infeasible solutions are usually good only for handling specific explicit constraints and maybe inefficient for implicit constraints. Furthermore, most repair strategies are problem domain specific.

The first method replaces the genotypes of infeasible solutions with their repaired counterparts. These repaired genotypes are then used to evaluate the fitness. This strategy is related to what is called *Lamarckian evolution* [90, 190]. Jean-Baptiste Lamarck (see Fig. 4.12 and [90]) was a botanist, zoologist and natural philosopher in France who assumed that an individual improves during its lifetime and that the resulting improvements are inheritated to its offsprings. The algorithmic implementation of Lamarckian evolution performs an improvement of the phenotype by any learning mechanism. Afterward the improvements are stored in the genotype.

Figure 4.12: Jean-Baptiste Lamarck ($\star$ 1st August 1744, † 18th December 1829).

The second method replaces the phenotypes (or temporarily modifies the genotype) of infeasible solutions without coding back the changes into the genotype. This strategy is related to a combination of learning and evolution, which is called the *Baldwin effect* [190, 183].

4.3.4.3 Penalizing Infeasible Solutions

Penalizing methods can be divided into four different approaches, namely:

Death penalty simply rejects all infeasible solutions. Problems can occur if the ratio p of feasible solutions is very small. Thus, it is possible that no feasible solution can be found by randomly generating candidate solutions.

Static penalty [74] uses the modified fitness function

$$\hat{f}(Indi) = \begin{cases} f(Indi), & \text{if } Indi^{\text{phenotype}} \text{ is feasible} \\ f(Indi) + \texttt{penalty}(Indi), & \text{otherwise,} \end{cases}$$

where $\texttt{penalty}(Indi)$ tends to zero (assuming a minimization problem) the less constraint violations occur. The $\texttt{penalty}(\cdot)$ function is usually based on a distance measure to provide a gradient to the feasible solution search space.

Dynamic Penalty [94] alters the penalty term by a factor which is proportional to the number of already computed iterations (generations) in such a way, that the penalty term becomes more and more relevant during evolution. One drawback of methods based on dynamic penalties is to predefine the annealing factor (and, if used, other constants).

Adaptive Penalty uses the idea, that if constraints pose no problem, the search should be performed with decreased penalties and vice versa. Thus, the parameters which influence the penalty term could depend for example on the ratio of feasible and infeasible solutions of the last k generations [10], or on some predefined thresholds, which define distances to feasible regions [170], or the parameters could be implemented self-adaptive as strategy parameters in the genotype.

4.3.5 Summary

This section provided fundamental information about two very important parts of EAs, namely the representation scheme and the evaluation functions. This information will be used as a starting point and as a motivation to introduce grammar based genotype-templates (in Sec.4.5) to define search spaces for EAs.

4.4 Evolutionary Operators

This section defines and discusses essential operators to modify the genotype representation. It is notable that there exist several problem specific evolutionary operators [115] like the virus infection approach [168, 146], multiple crossover schemes [193], order conserving permutation operations [138, 189, 144] and many others, which will not be mentioned here because of their specialized application areas. But all evolutionary operations could be classified into mutation, recombination or selection. The following inspection of different evolutionary operators uses an EC point of view and not a biological point of view, although again terms originated from biology are used. Evolutionary operations on tree based genotypes which are used for a concrete EA implementation in chapter 5 are presented in Sec.4.5.6.

4.4.1 Mutation

The *mutation* operator performs, depending on the implementation, changes in the variables of a genotype and/or changes in the structure of a genotype. These changes can be implemented in various ways, but a general definition of the mutation function, which is often called a *one-parent-operator*, because the mutation function takes only one genotype as input, can be given as follows.

Definition 4.1 (Mutation function). *The function* $\texttt{mut}(genotype(Indi), \underline{p}_{mut})$ *is a function which varies the variables or the structure of the genotype of Indi with a probability given by a mutation probability vector* $\underline{p}_{mut}$. *The output of* $\texttt{mut}(genotype(Indi), \underline{p}_{mut})$ *is the mutated individual* $In\hat{d}i$.

The different possible variations performed by the mutation function can be classified by the target(s) of the variation, namely:

- Variation of binary valued genotype variables, which is mostly implemented by inverting each binary value in the genotype with a given probability.
- Variation of integer valued genotype variables, which can be implemented by increasing/decreasing or randomly replacing of an integer value in the genotype with a given probability.
- Variation of real valued genotype variables, which can be done by increasing/decreasing or randomly replacing each real value in the genotype with a given probability. The variation of real valued genotype variables are often performed by using concepts developed for a special methology called evolutionary strategies.
- Variation of the genotype structure, which can be implemented by rearranging, shortening or widening the genotype structure.
- Variation by swapping, shifting, scrambling, inverting, etc. the content of several memory cells of the genotype. These operations are referred as sequencing-operations and are used in the field of solving permutation problems, where the ordering and non-repetition of values is important. Because this problem type does not occur in the context of this thesis, the interested reader is referred to [125].

In the methology of evolutionary strategies real valued genotype variables are divided into two different classes, namely *decision variables* and *strategy parameters*. The decision variables are used to form the phenotype and the strategy parameters are self-adjusting real valued variables parameterizing the mutation function.

Thus, each ES-individual is equivalent to a vector of real numbers and contains values for all decision variables $x_j \in \mathbb{R}(j = 1, 2, \dots, J)$ for the stated problem. Furthermore each individual contains $n_\sigma (1 \leq n_\sigma \leq J)$ standard deviations $\sigma_k \in \mathbb{R}^+ (k = 1, 2, \dots, n_\sigma)$ which are called (average) *rate of mutations*. These σ_k are strategy parameters which are adjusted self-adaptively during the optimization process. The decision variables of an offspring are inherited (via a recombination function - see next section 4.4.2) by one of the parents (same as in GAs), whereas the strategy parameters are inherited by intermediate crossover. Mutation is the main operator in ES and is done by changing the values of σ_j and x_j by two different methods. First the values σ_j are multiplied with a normal distributed random number. Then every decision variable x_j is changed by adding a normally distributed random number with expected value zero and standard deviation σ_j.

4.4.2 Recombination

The recombination operator is inspired by the principles of sexual reproduction in biology and thus utilizes parts of (at least) two parent-genotypes to build up a new genotype which is called *offspring-genotype* and is therefore also called *multiple-parent-operator*. Again, the implementation of recombination can be done in various ways and for tackling permutation problems, which are not focused in this thesis, special recombination implementations are necessary. Further information can be found in [138, 189, 144, 125]. Nevertheless, a general definition of a *recombination function* is as follows.

Definition 4.2 (Recombination function). *Denote by $\underline{Gen}$ a set of genotypes of individuals of population Pop, than $\texttt{rec}(\underline{Gen}, \underline{p}_{rec})$ is a function which utilizes parts of these genotypes to build up a new genotype which is denoted as offspring-genotype Gen^{rec}. The recombination probability vector $\underline{p}_{mut}$ gives the probability how much genotype-material each genotype shares with the resulting offspring-genotype*[4].

[4]More common is the use of a single probability value to determine the probability if recombination between certain parent-genotypes happens.

To illustrate the functionality of recombination lets discuss a widely used recombination technique called *n-point-crossover.* Consider the case of two string based genotypes (parent-genotypes) in which each memory cell contains a binary based variable. By selecting a number of string-position, referred as *crossover-points*, it is possible to recombine the two genotypes, resulting in a new genotype (i.e. offspring-genotype), as depicted in Fig. 4.13. If memory cells are copied randomly from

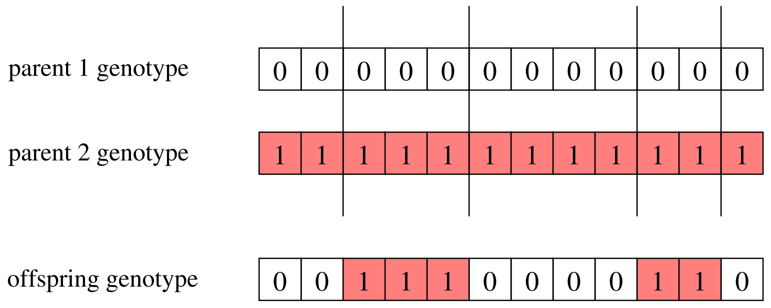


Figure 4.13: 4-point-crossover in strings-based genotypes with two parents. Only one of the two possible offspring structures is shown.

any parent the crossover scheme is called *uniform crossover.* Uniform crossover is mostly used in a parameterized form [177, 167], meaning that for each memory cell a given probability is used to decide if this or its counterpart from the other parent-genotype is used in the offspring-genotype. These crossover schemes can easily be extended to deal with more then two parent-genotypes, leading to so called *multi-recombination-crossover* . This multi-recombination techniques seems to have advantages since [49, 182, 2] reported that global optima where found faster and more often using multi-recombination then two-parent-recombination.

In genetic programming, where the individuals consist of unbalanced trees, crossover is implemented as subtree swapping (see Fig. 4.14). Multi-recombination can be implemented straightforward by swapping sub-trees between more than two parent individuals.

If $Gen_i = \mathtt{rec}(\underline{\tilde{Gen}}, \underline{p}_{rec})$ with $\forall \tilde{Gen}_i \in Pop_{t-1}$ holds for all $Gen_i (i = 1, \ldots, I)$ with I equal to the number of individuals in Pop_t the replacement method is called *generational replacement* , meaning that between each successive generations the complete population is replaced by recombined individuals. If at least one genotype

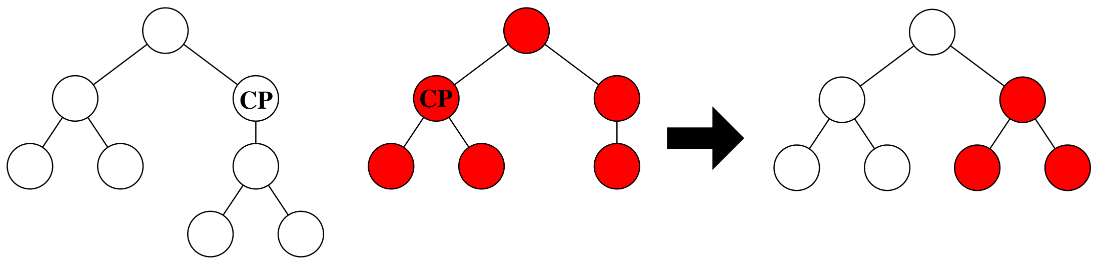


Figure 4.14: Crossover in tree-based genotypes. The two shown parent-genotypes exchange parts of their tree-structure at the randomly chosen crossover points (CP) . Only one of the two possible offspring structures is shown.

in Pop_t is not a recombination of genotypes of Pop_{t-1}, the replacement strategy is referred as *steady-state replacement* [177].

4.4.3 Selection

Selection is the operation by which individuals are chosen for reproduction. Reproduction means creating an offspring by recombination or cloning. Selected individuals are copied into the so called *mating pool* which can be seen as a temporary cache for individuals and is in the following referred as $Pop_{\text{mating pool}}$. Selection is performed on the basis of fitness values which are calculated with help of a so called fitness function (see Sec. 4.3.2). The fitness does not necessarily depend only on the phenotype of the individual, but can also be influenced by the genotype of the individual.

Definition 4.3 (Selection function). *Denote by* $< Indi >$ *a tuple of individuals of population Pop, than* $\texttt{sel}(f(< Indi >))$ *is a function which returns an Indi on the basis of the fitness values of* $< Indi >$.

Again, there exist various possible selection schemes, but the three most common are:

- Fitness-proportional selection.
- Rank-based selection.
- Tournament selection.

All these selection methods share the characteristic that they are *not extinctive*, meaning that even the individual with the worst fitness has a chance to reproduce. To compare different selection schemes [12] suggests to define some terms on the base of the *fitness distribution* of a population, which is defined as follows:

Definition 4.4 (Fitness distribution). *The function $f_{dis} : \mathbb{R} \rightarrow \mathbb{Z}_0^+$ assigns to each fitness value $f_{Indi} \in \mathbb{R}$ the number of individuals in a population Pop carrying this fitness value, where f_{dis} is called the fitness distribution of a population Pop.*

Comparable expressions based on the fitness distribution are:

Selection intensity: The term *selection intensity* was introduced in population genetics [19] to obtain a normalized and dimension-less measure and is defined as follows.

Definition 4.5 (Selection intensity). *The change of the average fitness of a population due to selection is called selection intensity and is calculated as*

$$SelInt = \frac{(\overline{f(Pop_{mating\ pool})} - \overline{f(Pop_t)})}{\sigma^*},$$

$$\begin{aligned} with \quad \overline{f(Pop)} &: \text{ average fitness of a population,} \\ \sigma^* &: \text{ mean variance of } f(Pop_{mating\ pool}). \end{aligned}$$

For completeness it should be noted that some authors [63, 6] use the term *selection pressure*, which also describes the change in the average fitness after applying a selection mechanism to the individuals of a population, computed on the base of the so called *takeover time*. Takeover has occurred if all individuals of a population have the same fitness value.

Selection variance: *Selection variance* is the expected variance of the fitness distribution of the temporarily population in the mating pool.

Definition 4.6 (Selection variance). *The selection variance is the normalized expected variance of the fitness distribution of the population after applying a* *selection method to the fitness distribution f_{dis}, i.e.*

$$SelVar = \frac{(\sigma^*)^2}{\sigma^2},$$
$$\text{with } \sigma^*: \text{ mean variance of } f(Pop_{mating\ pool}),$$
$$\sigma: \text{ mean variance of } f(Pop_t).$$

4.4.3.1 Fitness-Proportional selection

Let $E(Indi_i) = I \cdot p_s(Indi_i)$ (with I as population size and $p_s(Indi_i)$ as fitness-proportional selection probability) be the expectation value of possible participations of $Indi_i$ in each reproduction iteration. Good fitness-proportional selection methods[5], as the in Alg. 4.1 described *stochastic universal sampling selection* technique, are characterized by a minimal spread. Nevertheless, based on basic analysis and some empirical observations, proportional selection schemes seem to be very unsuited [12].

Algorithm 4.1 `Stochastic Universal Sampling`$(\cdot)$

Input: Pop_t

Output: $Pop_{\text{mating pool}}$.

```
(1)  $Pop_{\text{mating pool}} \leftarrow \varnothing$
(2)  $sum \leftarrow 0$
(3)  $pointer \leftarrow \texttt{drand}(0,1)$
(4)  for $i \leftarrow 1,\dots,I$
(5)    $sum \leftarrow sum + E(Indi_i)$
(6)    while $sum > pointer$
(7)      $Pop_{\text{mating pool}} \leftarrow Indi_i$
(8)      $pointer \leftarrow pointer + 1$
(9)    end while
(10) end for
```

[5]The classical *roulette wheel selection* have a high spread and is therefore no good choice.

4.4.3.2 Rank-based selection

In *rank-based selection* methods [188, 5], the population is sorted according to their fitness. The selection criteria for each individual depends therefore only on its position in the individuals rank and no longer on the actual fitness value. The probability of each individual being selected for mating is its ranking normalized by the population size. The commonly used rank-based selection method uses linear[6] ranking which lead to following selection probabilities p_s for $Indi_i$ to be copied into the mating pool. We have:

$$\begin{aligned} p_s(Indi_i) &= \frac{1}{I}\left(E_{\max} - (E_{\max} - E_{\min})\frac{r(Indi_i)-1}{I-1}\right), \qquad (4.2) \\ \text{with} \quad & p_s(Indi_i) \geq 0 \quad \forall i \in \{1,\dots,I\}, \\ & \sum_{i=1}^{I} p_s(Indi_i) = 1. \end{aligned}$$

Because of the constraints given in (4.2), $E_{\min} = 2 - E_{\max}$ and $1 \leq E_{\max} \leq 2$ must hold. Ranking introduces a uniform scaling across the population and provides a simple and effective way of controlling the selection intensity.

4.4.3.3 Tournament selection

In *tournament selection* [63] a number C of competitors (individuals), often denoted as *tournament size*, is chosen randomly from the population and the best individual from this group is allowed to reproduce. The parameter for tournament selection is the tournament size C with valid values ranging from $1,\dots,I$ (number of individuals in population). The selection intensity for tournament selection can approximately calculated [12] as

$$SelInt_{\text{Tour}}(C) \approx \sqrt{2\cdot(\log(C) - \log\sqrt{4.14\cdot\log(C)})} \qquad (4.3)$$

[6]For an example using non-linear ranking see page 60 in [125].

and the approximation for the selection variance is given by [12]

$$SelVar_{\text{Tour}}(C) = 1 - 0.0096 \cdot \log(1 + 7.11 \cdot (C-1)), \quad (4.4)$$
$$\text{with } SelVar_{\text{Tour}}(2) = 1 - \frac{1}{\pi}.$$

As already mentioned in [13]:

> ”It is shown that for the same selection intensity tournament selection has the smallest loss of diversity and the highest selection variance. It is concluded that tournament selection is in some sense the best selection method among the three” [7].

The aim of this section was to give a short outline about the functionality of the most common selection schemes and their pros and cons. The two important measures for recombination operations, namely selection intensity and selection variance were presented to provide the reader with information about the measures which are commonly used to predict the number of steps until a population converges to a unique solution. Concerning this thesis the main reason for this brief summary of recombination schemes was to justify the decision of the author to use tournament selection in the EA implementations presented later.

4.5 Tree Based Genotype Representation

This section provides the reader with the encoding scheme of candidate solutions, which will be used in the following chapters. The search space of the evolutionary algorithms will be defined by so called *genotype-templates*. This proposed novel and very general applicable concept of genotype-templates simplifies the design of problem specific genotype representations. The author describes how a genotype-template can easily be formulated with the help of grammars. This method should not be confused with genetic programming concepts, which also utilizes tree-based genotype representations, but which does not provide a grammar-based and simple scheme to define the genotype search space, nor the possibility to encode nearly arbitrary problems. The proposed method of genotype-templates combines the expressive power of grammars with the advantages of a general tree-based encoding scheme.

[7]They refer to tournament, truncation and ranking selection.

Especially by building genotypes from scratch, the challenge shifts from finding a solution for the original problem to the task of finding a possible and adequate genotype encoding. A good genotype representation should fulfill the following requirements:

- Simplicity in construction of variable length genotype templates.
- Easiness to implement evolutionary operations.
- Possibility to constrain genotype variables.
- Possibility to introduce expert knowledge.
- Implicit support of building blocks.

The usage of trees to represent genotypes is an elegant method for many encoding tasks, to solve, or at least an alternative, to simplify the fulfillment of these requirements[8]. To explain how the fulfillment of these requirements is supported by representing the genotype with tree structures, the next section provides some basics of trees. Referring to these basics Sec. 4.5.2 shows the qualitative potential of tree structures to fulfill the stated requirements.

4.5.1 Tree Basics

An interrelated, undirected and non-circular graph is referred as *tree*. Unbalanced trees are trees of data with any number of branches. A *node* is one branch of a tree and all nodes and connections of a tree will be referred as *complete-tree*. Lets A be a node with two branches (see Fig. 4.15), namely B and C, then B and C are called *child-nodes* of node A, and A is called *parent-node* of B and/or C. The only node in a tree without a parent-node is called *root-node* and nodes without child-nodes are called *leaf-nodes.* The *depth of a node* is equal to the number of parent-nodes on the graph to the root node plus one. Thus, a root-node has a depth of one. For the children of the root-node (i.e. B and C) the depth is two, and so on. Child-nodes with the same parent-node are called *siblings* and a *level* subsumes all nodes with the same depth. A *sub-tree* of a complete-tree subsumes all nodes starting at a certain node of a complete-tree. Thus, a complete-tree is a subtree of itself.

[8]The more astonishing is the fact that up to now there exist only very nascent attempts at extending EC theory to tree encodings [178, 139].

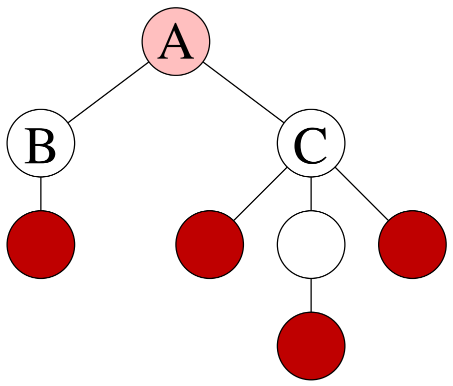


Figure 4.15: Exemplary unbalanced tree with eight nodes, a light red shaded root-node, four dark red shaded leaf-nodes and a total depth of four.

4.5.2 Tree Structures for Genotype Encoding

By using tree structures to fulfill the above stated requirements for a good genotype representation, the nodes of the used tree should also fulfill some requirements.

- Nodes can act as containers for variables (decision variables and/or strategy parameters).
- Classes of nodes are distinguishable by a node label.
- Nodes can be tagged as untouchable. Untouchable nodes (and their content) can not be removed or altered by evolutionary operations.
- Nodes can be tagged as inactive which means that the complete subtree with node as root-node will not be used in the genotype to phenotype mapping.

Each genotype representation needs a place to store information which is used to build the phenotype. Obviously the most common place are provided locations, as the base-pairs in the natural genotype or decision variables in artificial genotypes. Beside this information storage, the structure of the genotype itself can carry information. Thus, it is necessary to distinguish between *variable information* and *structure information*. The variable information in trees can be localized in nodes and each variable information location can be labeled with a variable name.

By regarding the structure of the genotype as information medium it is straightforward to subsume nodes with identical variable information locations by assigning them the same node-label. By labeling the nodes it is possible to construct a genotype state space with the help of a grammar [160]. In the following a genotype state space description is denoted as *genotype-template*.

4.5.3 Example of a Tree Based Genotype Representation

To clarify this concept let us construct a simple example. Assuming the task is to find a (sub)optimal input vector $\underline{x}$ consisting of three elements to minimize a system output y. Each vector element could lie in the interval $[-10, 10]$. Figure 4.16 illustrates a

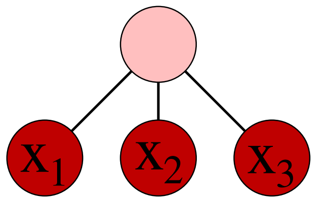


Figure 4.16: Tree based genotype representation.

possible encoding with trees. The root node can be labeled as ”VECTOR” and each node-children of the root node can be labeled as ”VECTOR-ELEMENT”. The nodes ”VECTOR-ELEMENT” are containers for the decision variables x_i.

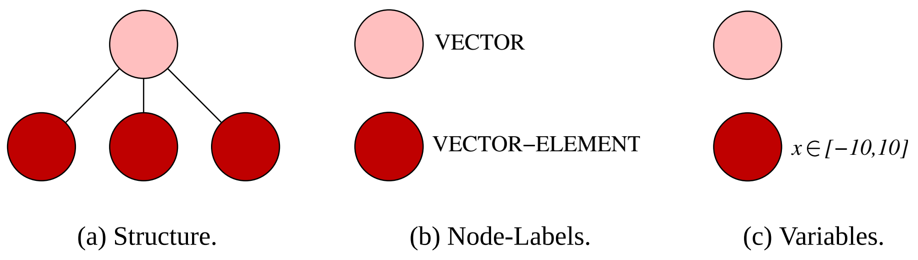


Figure 4.17: Simple example of a tree based genotype representation with (a) illustrating the structure, (b) the corresponding node-labels and (c) the embedded variables.

A more informative illustration of the same genotype is given in Fig. 4.17, where the interesting information about the genotype representation is divided into three sub-figures, giving information about the tree structure, node-labels and decision variables.

4.5.4 Grammars as Framework for Genotype-Templates

By using node-labels it is possible to define the space of all possible genotypes with help of a grammar, where a grammar is a high-level notation used to describe the structure of data, for example the structure of a sentence or the structure of a genotype. Consider the following grammar 4.1 that defines e.g. the structure of (a subset of) English sentences:

Grammar 4.1 Grammar that defines the structure of a subset of English sentences.

<sentence> ::= <nounphrase> <verbphrase>.
<nounphrase> ::= <article> <noun>
<verbphrase> ::= <verb> <nounphrase>
<article> ::= **a** | **the**
<noun> ::= **man** | **dog**
<verb> ::= **likes** | **bites**

Grammar 4.1 is in *Backus-Naur form* (BNF) [8] , which is a way of representing context free grammars. BNF was developed in the 1960s by the ALGOL committee, in particular, John Backus and Peter Naur, to describe the ALGOL programming language. It is equivalent in descriptive power to context free grammars developed independently by Noam Chomsky to describe natural languages. A BNF grammar is a set of rules and a single non-terminal (called the start symbol). A rule has:

- Non-terminals, enclosed in <>, which must be defined by (appear by itself on the left hand side of) at least one rule.
- Terminals, actual strings like "." and the words in the example (e.g. **dog** in grammar 4.1).
- Metasymbols like ::=, |,., <>.

By having the requirements for nodes in a tree for genotype encoding in mind (see Sec. 4.5.2), it is possible to formulate a grammar which defines the genotype space based on trees.

Note that in grammar 4.2 the symbols boolean, integer, double and string are stated only for the sake of simplicity as terminal symbols. Obviously these symbols

Grammar 4.2 Grammar defining a genotype space on the basis of trees.

<genotype-template> ::= <node-list>,<conjunction-list>,<constraint-list>

<node-list> ::= **empty** | <node>,<node-list>
<node> ::= <node-label>,<min-succ-size>,<max-succ-size>,<var-list>
<conjunction-list> ::= **empty** | <conjunction>,<conjunction-list>
<conjunction> ::= <node>,<node>,<predetermined>
<var-list> ::= **empty** | <var>,<var-list>
<var> ::= <var-name>,<min-value>,<max-value>,<var-type>,<untouchable>
<constraint-list> ::= **empty** | <constraint>,<constraint-list>
<constraint> ::= <node>, <var-name>,<condition>,<scope>

<predetermined>,<untouchable> ::= **boolean**
<min-succ-size>,<max-succ-size> ::= **integer**
<min-value>,<max-value> ::= **double**
<node-label>,<var-name>,<var-type> ::= **string**
<condition> ::= > | < | **==** | **!=**
<scope> ::= **sibling** | **level** | **all**

are again defined by non-terminal symbols (to be precise this should be done by some regular expression).

With the non-terminal symbol <untouchable> it is possible to detach certain decision variables embedded in the concerning nodes from evolutionary operations like mutation. The non-terminal symbol <predetermined> is used to force specific compounds to be present in the genotype. By using the grammar 4.2, the genotype of the example given in Sec. 4.5.3 can easily be described by the following genotype-template:

Genotype Template 4.1 Genotype template for example given in Sec. 4.5.3.

`node` (VECTOR, 3, 3, empty)
`node` (VECTOR-ELEMENT, 0, 0, *x*)
`var` (*x*, -10.0, 10.0, real, FALSE)
`conjunction` (VECTOR, VECTOR-ELEMENT, FALSE)

4.5.5 Constraint Handling in Tree Based Genotypes

Most problems which are tackled with methods of EC exhibit several problem specific constraints. As mentioned in Sec. 4.3.4 there exist three possible methods to deal with constraints, namely avoiding infeasible solutions, repairing infeasible solutions and penalizing infeasible solution. By using grammar based genotype-templates many constraints can already be fulfilled by setting up a problem specific genotype-template. For example the trivial constraint that a vector should consist of exactly three elements is implicitly given by the concerning genotype-template 4.1. This kind of requirements is already checked by performing the evolutionary operations. Thus, infeasible solutions are avoided.

In grammar 4.2 the attentive reader should have recognized the existence of a non-terminal symbol referred as <constraint>. By using this symbol it is possible to construct simple constraints like an increasing order of decision variable values. The scope where this configuration should be fulfilled is given by the non-terminal symbol <scope>, which can have as values **sibling**, **level** or **all**. In the genotype-template 4.1 a statement like `constraint` (VECTOR-ELEMENT, *x*, >, **sibling**) forces at certain stages of the evolutionary process a fulfillment of the claimed alignment. Because the unwanted configuration of the decision variables are sometimes present in the genotype, this constraint handling method can be classified as repairing infeasible solutions.

4.5.6 Evolutionary Operations on Tree Based Genotypes

With respect to common genotype representations evolutionary operations on tree based genotypes differ only for the mutation operation and for the recombination operation, which will be pointed out in the following.

4.5.6.1 Mutation of Tree Based Genotypes

By considering mutation operations we have to distinguish between variable information mutation and structure information mutation.

The mutation of variable information is implemented by traversing all nodes of the tree based genotype representation. Each node is checked for variables and if there are variables each variable is, due to a certain probability, chosen to be subject of mutation. Values of real decision variables are changed by means of evolutionary strategy concepts as described in Sec. 4.4.1. Values of integer decision variables are changed by increasing, decreasing or replacing the original value by chance. The grammar based genotype-template provides in all cases the legal ranges of each variable.

The mutation of structure information is implemented by adding, deleting or expanding a randomly chosen node. Again, the grammar based genotype-template provides all information to validate the correctness of each structure information mutation.

4.5.6.2 Recombination of Tree Based Genotypes

Compared to string based genotypes the recombination operations simplifies to sub-tree swapping. One big advantage of tree based genotypes is that the tree structure inherently offers a classification of functional entities represented by sub-trees. Furthermore, in the case of grammar based genotype-templates the validation if the sub-tree swapping results in valid genotypes can be easily and efficiently performed by checking the grammar.

4.6 Summary

This chapter introduced the main concepts of EC and outlined the importance of the genotype representation for each EA. Furthermore the advantages and disadvantages of different selection schemes was shown. It was clarified that the search space of the evolutionary process is always defined by the candidate solution encoding and thus, a good genotype representation is the core of each EA. Furthermore it was stated that in most applications the challenge shifts from finding a solution for the original problem to the task of finding a possible and adequate genotype encoding. Because of this, trees were introduced to encode candidate solutions with the demands to simplify constraint handling, accelerate implementation and to provide the capability to handle variable length encodings. The most important subject matter in this chapter was the introduction of a novel concept, which uses grammars as tool to formulate general usable genotype-templates.

Chapter 5

Evolutionary Optimization of Descriptive Takagi-Sugeno Fuzzy Models

This chapter provides a grammar based genotype-template which is used to construct highly interpretable DTSFM candidates with a partly defined rule-base (Sec. 3.3.4). These DTSFM candidates are used in an evolutionary process which directs the search to optimal DTSFMs. The genotype-template and the algorithmic description of the evolutionary loop provided in this chapter are used in chapter 6 to model an artificial and a complex real world system.

5.1 Michigan vs. Pittsburgh Approach

There exist two commonly used methods to represent FMs in a population. The so called Michigan approach [26] utilizes each individual in a population as one part (i.e. rule) of the candidate solution. Thus, the complete population of each generation represents only one candidate solution. Each individual can be seen as a functional entity of the overall candidate solution. For many candidate solutions it is meaningful to introduce more than one type of functional entity. In this case the Michigan approach has to deal with sub-populations, with each sub-population consisting of individuals representing the same type of functional entity. The hardest problem that occurs using the Michigan approach is to find an optimal combination scheme for the individual to form a candidate solution.

In the second approach, known as the Pittsburgh approach [27], each individual represents a candidate solution. Thus, the complete population of each generation represents several candidate solutions. In combination with tree based genotypes the Pittsburgh approach is obviously more suitable to use. This is caused by the fact that each sub-tree can already be seen and encoded as functional entity. The difficult problem of merging functional entities to a candidate solution (which is again a highly non-linear problem) does not occur, because it is assumed to be done by the evolutionary operations like crossover.

5.2 Acquiring the Genotype Tree Structure

It is useful to sketch the structure of the tree representation of the target genotype-template (see Fig. 5.1), which will than be formulated with help of grammar 4.2. During modeling the candidate DTSFMs should be able to make use of different features from an arbitrarily large feature set. These features should be covered with b-splines which act as fuzzy sets.

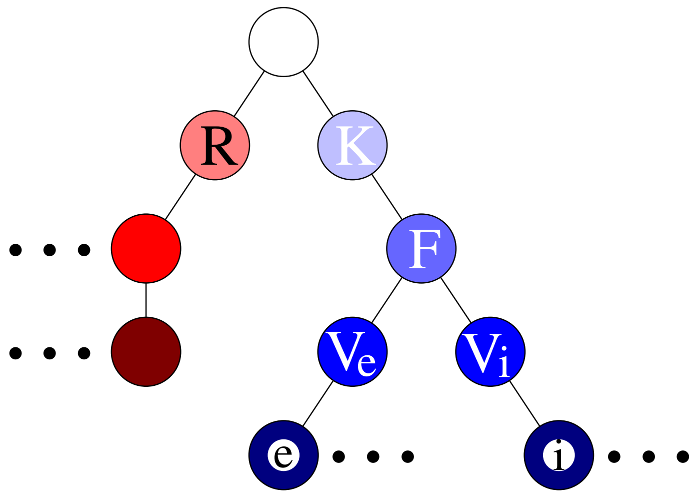


Figure 5.1: Sketch of the target tree based genotype. The concerning genotype template is given in grammar 5.1
.

5.2.1 Rule-Base and Knowledge-Base

The structure of a FM consist of two main parts, the rule-base and the knowledge-base. These two parts are also reflected in the sketch of the genotype tree of Fig. 5.1.

The left side encodes the rule-base and the right side encodes the knowledge-base. Below the starting node of the rule-base there are the single rules, each rule containing one or more premises. The left hand side of the tree represents the knowledge-base, with single features (inputs) of the problem, each covered by a knot-vector which constitutes the b-spline based fuzzy-sets.

5.2.2 Genotype to Phenotype Mapping

By using genotype-templates the implementation of problem specific genotypes becomes much more efficient with respect to implementation and maintenance time of the resulting evolutionary algorithm. But it has always to be kept in mind that the function, which performs the genotype to phenotype mapping, can be far from trivial and still has to be carried out by an expert. Regarding this genotype to phenotype mapping function for DTSFMs some specific considerations have to be explained.

5.2.3 B-spline Specific Implementation Considerations

Because of implementation reasons the knot-vector was split into an external and an internal part, each with knot-positions as elements. The external knot-vector consist of two x knots, with $x = 2 \cdot k$ (k = order). Thus, for example, six b-splines of order three are defined over an external knot-vector consisting of six knots (three left external knots plus three right external knots) plus three internal knots as illustrated in Fig. 5.2. This terminology differs from the commonly used one [38], but it is very convenient for our implementation.

It is convenient, because it is desirable to encode the genotype knot-positions with possible values in a fixed range (e.g. $[0,1]$). Because of interpretability reasons, it is also desirable that the b-splines form a partition of unity in the respective input dimension. By rescaling[1] the internal knot-position to (assuming zero as minimum and one as maximum) to the minimum (min_n) and maximum (max_n) values of the concerning input dimension, it is always guaranteed that the complementarity condition hold. The encoded external knot-vector consist one half each of the left external knot-vector $\underline{\lambda}^{\text{left}}$ and right external knot-vector $\underline{\lambda}^{\text{right}}$, both with possible values in $[0,1]$ and number of elements equal to the order k.

[1] In the following the term rescaling is used to describe the rescaling during the genotype to phenotype mapping.

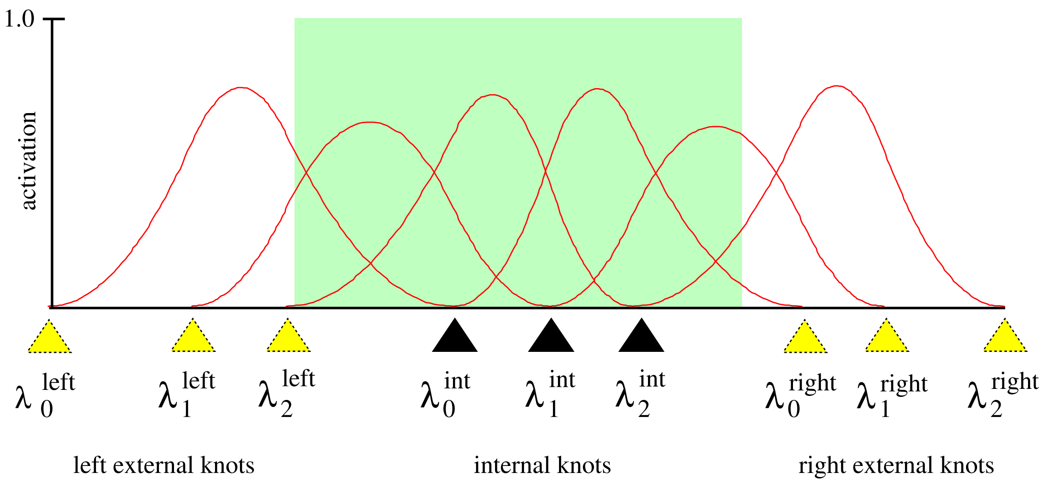
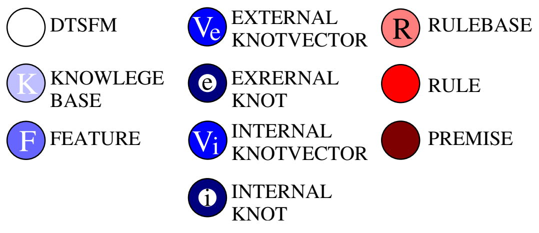


Figure 5.2: Left external, internal and right external knot-vector forming a combined knot-vector $\underline{\lambda} = (0, 1.2, 1.99, 3.4, 4.2, 4.9, 6.3, 7, 8)$ with an assumed input interval [2,6]. The green shaded area represents this interval in which the internal knots can be moved, new knots can be added or old knots can be deleted. The external knots are always arranged outside this interval.

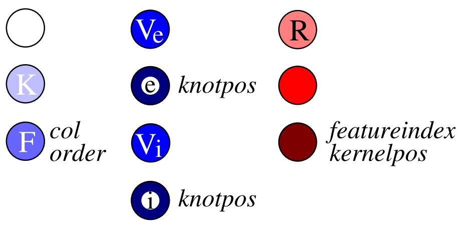
The encoded left external knot-vector $\underline{\lambda}^{\text{left}}$ is rescaled (assuming zero as minimum and one as maximum) to $min_n - \frac{(max_n - min_n)}{4}$ (leftmost possible knot-position of $\underline{\lambda}^{\text{left}}$) and $min_n - \frac{(max_n - min_n)}{1000}$ (rightmost possible knot-position of $\underline{\lambda}^{\text{left}}$). The encoded right external knot-vector $\underline{\lambda}^{\text{right}}$ is rescaled (assuming zero as minimum and one as maximum) to $max_n + \frac{(max_n - min_n)}{1000}$ (leftmost possible knot-position of $\underline{\lambda}^{\text{right}}$) and $max_n - \frac{(max_n - min_n)}{4}$ (rightmost possible knot-position of $\underline{\lambda}^{\text{right}}$).

5.2.4 Feature-Set Selection Implementation Considerations

A decision variable called *featureindex*, which is located in each PREMISE node of a RULE node, acts as a selector which of the encoded features in the knowledge-base is used. Because the knowledge-base encodes between ***coded min rule*** and ***coded max rule*** (predetermined by the model designer) rules, the decision variable should be able to accept values in the range [***coded min rule***, ***coded max rule***]. Furthermore the decision variable *featureindex* should be be assigned only distinct values for all siblings. Figure 5.3 illustrates the node labeling and the decision variables which are embedded in the nodes.



(a) Labels.



(b) Variables.

Figure 5.3: Genotype labels and variables of a DTSFM.

5.2.5 Fuzzy-Set Selection Implementation Considerations

The decision variable *kernelpos* can store real values in the range [0,1]. These values are converted by Alg. 5.1 to indices of fuzzy sets (defined in the knowledge-base) covering the concerning input. During the phenotype to genotype conversion, *kernelpos* (assuming zero as minimum and one as maximum) is rescaled to the minimum available data-pattern value of the concerning input (min_n) and to $max_n - \frac{(max_n - min_n)}{4}$, where max_n is the maximum available data-pattern value of input n.

Algorithm 5.1 Kernel selection algorithm.

Input: Genotype decision variable *kernelpos*.

Output: Index i of concerning fuzzy set.

```
(1) $n \leftarrow col$
(2) $kernelpos \leftarrow \texttt{rescale}(kernelpos)$
(3) $stop \leftarrow$ FALSE
(4) $i \leftarrow order_n$
(5) while $i <$ number of elements$(\underline{\lambda}^n)$ and $stop$ = FALSE
(6)     if $kernelpos < \underline{\lambda}_i^n$ then $stop \leftarrow$ FALSE
(7)     $i \leftarrow i + 1$
(8) end while
(9) $i \leftarrow i - (order_n + 1)$
```

5.3 The Used Genotype-Template

The on grammar 4.2 based syntactic formulation of the so far described genotype-template is given by template 5.1. This template contains only very few parameters which have to be predetermined by the user. The bold-and-italic typeset symbols

Genotype Template 5.1 DTSFM genotype template.

```
node (DTSFM, 2, 2, empty)
node (KNOWLEDGE BASE, max feature, max feature, empty)
node (FEATURE, 2, 2, (col, order))
node (EXTERNAL KNOT-VECTOR, max order · 2, max order · 2, empty)
node (INTERNAL KNOT-VECTOR, min kernel, max kernel, empty)
node (EXTERNAL KNOT, 0, 0, knot pos)
node (INTERNAL KNOT, 0, 0, knot pos)
node (RULE BASE, min coded rule, max coded rule, empty)
node (RULE, min premise, max premise, empty)
node (PREMISE, 0, 0, (featureindex, kernel pos))

var (col, 0, max col −1, integer, FALSE)
var (order, min order, max order, integer, FALSE)
var (knot pos, 0.0, 1.0, real, FALSE)
var (featureindex, 0, max feature −1, integer, FALSE)
var (kernel pos, 0.0, 1.0, real, FALSE)

conjunction (DTSFM, RULE BASE, TRUE)
conjunction (DTSFM, KNOWLEDGE BASE, TRUE)
conjunction (KNOWLEDGE BASE, FEATURE, FALSE)
conjunction (FEATURE, EXTERNAL KNOT-VECTOR, TRUE)
conjunction (FEATURE, INTERNAL KNOT-VECTOR, TRUE)
conjunction (EXTERNAL KNOT-VECTOR, EXTERNAL KNOT, FALSE)
conjunction (INTERNAL KNOT-VECTOR, INTERNAL KNOT, FALSE)
conjunction (RULE BASE, RULE, FALSE)
conjunction (RULE, PREMISE, FALSE)

constraint (FEATURE, col, ! =, sibling)
constraint (EXTERNAL KNOT, knot pos, >, sibling)
constraint (INTERNAL KNOT, knot pos, >, sibling)
constraint (PREMISE, featureindex, >, sibling)
```

represent external variables which have to be predefined by the model designer or are derived from the given data-set (i.e. ***max col*** = number of columns of the input matrix). In fact only four different model predicates have to be given, namely the maximum number of features the EA is allowed to select from the input data (i.e. ***max feature***), ranges for the smoothness of the used fuzzy sets (e.g. ***min/max order***), ranges for the used number of fuzzy sets covering each input (i.e. ***min/max kernel***) and ranges for the encoded number of rules (i.e. ***min/max coded rule***). Keep in mind that the number of rules of the phenotype can be lower than the number given by ***min coded rule***, because it is possible that the genotype contains several equal rules, of which only one instance is used in the resulting model.

5.4 The Implemented Evolutionary Algorithm

This section provides an explanation of the used EA in pseudo-code. The input parameters of the evolutionary process are firstly the instructions how to encode the problem, i.e. the genotype-template, secondly standard evolutionary parameters like $PopulationSize$ and $TournamentSize$, and thirdly parameters ***min/max order*** concerning the target phenotype, i.e. concerning a DTSFM.

Regarding the first point grammar 4.2 will be used. Regarding the second point the evolutionary parameters comprise normally also parameters like mutation and crossover probability. To keep the approach as simple as possible these parameters were fixed for all runs which are performed to achieve the results presented in chapter 6. Only the $PopulationSize$ and the stop criteria varied for the artificial and the real world data-set. The other reasonable evolutionary parameters which are implemented as parts of the overall EA are described in the following sections.

5.4.1 Implementation Environment

Obviously a detailed description of all implemented functions would go beyond the scope of this thesis; but for information, the described algorithms are embedded in a software library programmed by the author of this thesis. The core library provides structures and functions to deal with matrices. Based on this, a generalized basis function network library and an evolutionary computation library were programmed. The used programming language is C with additional functionality given by the GLib [60]. The GLib is a library which provides, besides many other things like definitions for standard variable types, functions to deal with trees. It is also used as core for GTK+ [68], a library for the designing of graphical user interfaces. Both, GLib and GTK+ are available for many platforms and both are distributed under the lesser general public license, which means that they are freely available for personal and commercial use.

5.4.2 Implemented Crossover

Crossover is implemented by randomly choosing two genotypes. Two nodes of the same type (identical node-labels) are selected, again by random, from each of these genotypes. In the first genotype the selected node (and the concerning sub-tree) is replaced by a copy of the sub-tree chosen in the second genotype. This is done each generation *IndiSize* times. Algorithm 5.2 gives a description in pseudo-code.

Algorithm 5.2 `crossover`(·).

Input: *pop*.
Output: Recombined *pop*.

```
(1)  for $i \leftarrow 1,\dots,IndiSize$
(2)      $r \leftarrow \texttt{drand}(1, IndiSize)$
(3)      $genotype^{\text{father}} \leftarrow genotype_r$
(4)      if $\texttt{drand}(0,1) < 0.5$
(5)          $r \leftarrow \texttt{drand}(1, IndiSize)$
(6)          $genotype^{\text{mother}} \leftarrow genotype_r$
(7)          $node^{\text{father}} \leftarrow RamdomNode(genotype^{\text{father}})$
(8)          $node^{\text{mother}} \leftarrow RamdomNodeOfType(genotype^{\text{father}}, node^{\text{father}})$
(9)          $SubtreeCopy \leftarrow \texttt{CopySubtree}(node^{\text{mother}})$
(10)         $node^{\text{father}} \leftarrow SubtreeCopy$
(11)     end if
(12) $NewPop \leftarrow NewPop + genotype^{\text{father}}$
(13) end for
(14) $pop \leftarrow NewPop$
```

5.4.3 Implemented Structure Information Mutation

The implemented structure mutation operation is described by Alg. 5.3. Keep in mind that the genotype-template can be used to countercheck if a node can be added or a node and the concerning sub-tree can be erased from the genotype. The functions `GetRandomNodeOfGenotype`$(\cdot)$ and `AddPossibleSubtree`$(\cdot)$ in Alg. 5.3 are assumed to perform this check and therefore, shrinking or growing of the genotype is only done in a space defined by the genotype-template. In addition to that the following algorithms make use of some basic functions with self-explanatory names like `drand`(min, max), which returns a random value between *min* and *max*. Instructions in row (7) and (10) of Alg. 5.3 implement the idea that the structural mutation should have, to certain degree, a general direction in terms of shrinking or growing the genotype-size. In this case the initial value of *DelProb* in instruction (3) causes a tendency to add new parts to the genotype.

Algorithm 5.3 `MutateStructureInfo`$(\cdot)$.

Input: *genotype*.
Output: Mutated *genotype*.

```
 (1) $NodeSize \leftarrow \texttt{GetNumberOfNodes}(genotype)$
 (2) $NodeSize2Mutate \leftarrow \texttt{round}(\sqrt{(NodeSize \cdot \texttt{drand}(0,1)^3})$
 (3) $DelProb \leftarrow 0.4$
 (4) for $i \leftarrow 1, \ldots, NodeSize2Mutate$
 (5)     $node \leftarrow \texttt{GetRandomGenotypeNode}(genotype)$
 (6)     if $\texttt{drand}(0,1) < DelProb$
 (7)         $DelProb \leftarrow \texttt{MIN}(0.95, DelProb \cdot 1.2)$
 (8)         $genotype \leftarrow \texttt{DelSubtree}(genotype, node)$
 (9)     else
(10)         $DelProb \leftarrow \texttt{MAX}(0.05, DelProb \cdot 0.8)$
(11)         $genotype \leftarrow \texttt{AddPossibleSubtree}(genotype, node)$
(12)     end if
(13) end for
```

5.4.4 Implemented Variable Information Mutation

The variable mutation operation is outlined in Alg. 5.4, which performs mutation of real valued decision variables by means of evolutionary strategies and mutation of integer valued decision variables by increasing, decreasing or random replacement of the original value. The decision variables are named v, the coupled strategy parameter is referred as v^{sp} and the in the genotype defined range for the decision variable v is given by v_{min} and v_{max}.

Algorithm 5.4 `MutateVariableInfo`$(\cdot)$.

Input: *genotype*.
Output: Mutated *genotype*.

```
 (1) $NodeSize \leftarrow$ GetNumberOfNodes($genotype$)
 (2) for all nodes $n$ in $genotype$
 (3)     if drand$(0,1) <$ drand$(0,0.05)$
 (4)         randomly reinitialize all decision variables in $n$
 (5)     else
 (6)         for all decision variables $v$ in $n$
 (7)             if type$(v) = real$
 (8)                 MutateRealValuedVariable$(v, v^{sp})$          (Alg. 5.5)
 (9)             end if
(10)             if type$(v) = integer$
(11)                 MutateIntegerValuedVariable$(v)$               (Alg. 5.6)
(12)             end if
(13)         end for
(14)     end if
(15) end for
```

Algorithm 5.5 `MutateRealValuedVariable`$(\cdot)$.

Input: v, v^{sp}

Output: Mutated v, v^{sp}.

```
(1)  $r \leftarrow \texttt{MAX}(\texttt{drand}(0,1), 0.000001)$
(2)  $v^{sp} \leftarrow v^{sp} \cdot \exp(0.1 \cdot \sqrt{-2 \cdot \log(r)} \cdot \sin(6.283185307 \cdot \texttt{drand}(0,1)))$
(3)  if $v^{sp} > v^{sp}_{max}$ or $v^{sp} < v^{sp}_{min}$
(4)      $v^{sp} \leftarrow \texttt{drand}(v_{min}, v_{max})$
(5)  end if
(6)  $r \leftarrow \texttt{MAX}(\texttt{drand}(0,1), 0.000001)$
(7)  $v \leftarrow v + v^{sp} \cdot \sqrt{-2 \cdot \log(r)} \cdot 0.001$
(8)  if $v > v_{max}$ or $v < v_{min}$
(9)      $v \leftarrow \texttt{drand}(v_{min}, v_{max})$
(10) end if
```

Algorithm 5.6 `MutateIntegerValuedVariable`$(\cdot)$.

Input: v.

Output: Mutated v.

```
(1)  if $\texttt{drand}(0,1) < 0.005$
(2)      if $\texttt{drand}(0,1) < 0.5$
(3)          if $\texttt{drand}(0,1) < 0.5$
(4)              $v \leftarrow v + 1$
(5)          else
(6)              $v \leftarrow v - 1$
(7)          end if
(8)      else
(9)          $v \leftarrow \texttt{round}(\texttt{drand}(v_{min}, v_{max}))$
(10)     end if
(11)     if $v > v_{max}$ or $v < v_{min}$
(12)         $v \leftarrow \texttt{round}(\texttt{drand}(v_{min}, v_{max}))$
(13)     end if
(14) end if
```

5.4.5 Implemented Evolutionary Loop

Algorithm 5.7 briefly outlines all steps that are performed during the evolutionary process to identify an optimal DTSFM. The stop criteria of this evolutionary loop could be a fitness value, a generation index or elapsed time. Note that in this implementation an individual (model) is the better, the smaller the fitness value becomes. This sounds irritating but it simplifies the fitness calculation and is an often used procedure. During the evolutionary process the fittest genotype is always stored as elite genotype. In the implementation the elite genotype has the index zero ($Indi_0$).

5.4.5.1 Initialization of the Genotype

The initialization in instruction (1) of Alg. 5.7 is done by building *IndiSize* genotypes with help of the genotype-template 5.1. The creation of each genotype starts always with the root-node. Subsequently valid nodes are added at random and filled with decision variables. The decision variables are also initialized randomly but within valid ranges provided by the genotype-template. The implemented initialization supports ”small” structures which will be subsequently driven to bigger structures. This initialization procedure is motivated by the fact that very special and narrow support areas can cause a failure in the parameter optimization process. This happens, for example, if not enough data-patterns lie in the support areas of the rules and the overdetermined system of linear equation can not be solved.

5.4.5.2 Generalization Error Estimation as Fitness Factor

The DTSFM fitness is proportional to the normalized mean square error of the computed model output, computed with help of Eq. (3.19), to the desired output. The model output is calculated by a function referred as `CvModelOutput`$(\cdot)$ (see instruction (15) in Alg. 5.7), which indicates that a cross-validation (see Sec. 2.6.3) is performed. By assuming a 10 times cross-validation ($cv = 10$) the cross-validation is done in the following way. After the genotype to phenotype mapping the model parameters are determined cv times on the base of $\frac{(cv-1)}{cv}$ available training data. Each time the model output is calculated without using the data that were used for parameter optimization. These outputs are added to the total model output vector. Because numerous different splittings of the dataset are possible, it is not a complete cross-validation. On the other hand the leave-one-out method, the computationally cheapest complete cross-validation, tends to include unnecessary components in the model, and has been shown to be asymptotically incorrect [174]. Other authors also showed that the leave-one-out method underestimates the true predictive error [119] and does not work well for data with strong clusterization [50].

To keep the golden mean, a cross-validation very similar to Monte Carlo cross-validation (MCCV) [171] was implemented. In MCCV the data are partitioned M times into disjoint train and test subsets, where the test subset is a fraction $\beta = \frac{(cv-1)}{cv}$ of the overall data [20]. The main difference between MCCV and normal cross-validation is that in MCCV the different test subsets are chosen randomly and need not be disjoint.

In this thesis the subsets are also chosen randomly for each complete cross-validation estimation, but they are disjoint. Every generation the generalization error estimation of the elite individual is recalculated as an average of the prior and the actual cross-validation estimation (see instruction (23) in Alg. 5.7). This procedure decreases the probability that a small model generalization error estimation is based on a disadvantageous and seldom subset selection. On the other hand every generation it is possible that a new model with a ”very good” generalization error estimation, possibly based on a disadvantageous subset selection, replaces the elite individual. To hold down the probability of these undesired occurrences the subsets are forced to be disjoint.

5.4.5.3 Fitness Penalization by Interpretability Factors

As discussed the interpretability of the resulting model can be a crucial aspect if, for example, the user wants to extract knowledge out of the model. By using b-splines as fuzzy sets and a complete rule-base with ”don’t care” premises only the number of possible fuzzy sets on each input, the number of ”non-don’t care” premises and the number of simultaneously activated rules are of interest[2]. The complementarity condition implicitly holds for b-spline based fuzzy sets. The first two interpretability factors can be fulfilled by an appropriate setting of the model parameters (e.g. the variables ***min/max kernel*** and ***min/max premise*** in the genotype-template 5.1). Thus, only the number of simultaneously activated rules is calculated in the evolutionary loop, where `IntFactor3`$(\cdot)$ of instruction (16) in Alg.5.7 implements Eq. (3.15).

[2]Until chapter 7 the ”leveling” interpretability factor IF_4 (Sec. 3.15) will not be discussed.

Algorithm 5.7 `EvolutionaryLoop`(·)

Input: Genotype Template, evolutionary parameters, DTSFM parameters.
Output: Evolutionary optimized *DTSFM*.

```
(1)  $pop \leftarrow \texttt{InitPop}(GenotypeTemplate, IndiSize)$
(2)  $fitness_{\text{elite}} \leftarrow \text{MAXDOUBLE}$
(3)  $generation \leftarrow 0$
(4)  while $StopCriteria = \text{FALSE}$
(5)      $fitness_{\text{BestInGen}} \leftarrow \text{MAXDOUBLE}$
(6)      if $generation > 0$
(7)          $pop \leftarrow \texttt{crossover}(pop)$                                  (Alg. 5.2)
(8)      end if
(9)      for $i \leftarrow 0, \ldots, IndiSize$
(10)         if $genotype_i > 0$ and $generation > 0$
(11)             $genotype_i \leftarrow \texttt{MutateStructureInfo}(genotype_i)$        (Alg. 5.3)
(12)             $genotype_i \leftarrow \texttt{MutateVariableInfo}(genotype_i)$         (Alg. 5.4)
(13)         end if
(14)         $DTSFM_i \leftarrow \texttt{genotype2phenotype}(genotype_i)$               (Sec. 5.2.2)
(15)         $nmse_i \leftarrow \texttt{NMSE}(\texttt{CvModelOutput}(DTSFM_i), DesiredOutput)$
(16)         $interpretability_i \leftarrow \texttt{IntFactor}_3(DTSFM_i)$              (Eq. 3.13)
(17)         $fitness_i \leftarrow nmse_i + \frac{1}{interpretability_i}$                 (Sec. 4.3.4.3)
(18)         if $fitness_i < fitness_{\text{BestInGen}}$
(19)             $fitness_{\text{BestInGen}} \leftarrow fitness_i$
(20)             $genotype_{\text{BestInGen}} \leftarrow genotype_i$
(21)         end if
(22)     end for
(23)     $fitness_{\text{elite}} \leftarrow \frac{(fitness_{\text{elite}} \cdot (age_{\text{elite}} - 1) + fitness_{\text{elite}})}{age_{\text{elite}}}$
(24)     if $fitness_{\text{BestInGen}} < fitness_{\text{elite}}$
(25)         $fitness_{\text{elite}} \leftarrow fitness_{\text{BestInGen}}$
(26)         $genotype_{elite} \leftarrow genotype_{BestInGen}$
(27)     end if
(28)     $pop \leftarrow \texttt{TournamentSelection}(pop, TournamentSize)$            (Sec. 4.4.3.3)
(29)     $generation \leftarrow generation + 1$
(30) end while
(31) $DTSFM \leftarrow \texttt{genotype2phenotype}(genotype_{elite})$                   (Sec. 5.2.2)
```

5.5 Summary

This chapter presented the description of the complete EA implementation for DTSFM identification. First of all the decision to use the Pittsburgh approach for encoding was justified. This was followed by presenting a possible genotype tree structure and the concerning genotype-template. The template was formulated on the base of grammar 4.2. After this the most important parts of the EA were described in pseudo-code.

At this stage all required parts and information for data-driven modeling via EC is provided and thus, in the next chapter the described implementation is tested on an artificial and on a real world dataset.

Chapter 6

Data Analysis

In this chapter the developed concepts and algorithms will be applied to an artificial and to a complex real world problem. The EA implementation as described in chapter 5 is used for both datasets.

6.1 Artificial Data

The purpose of this section is to validate the in the last chapter provided EA. By using an artificial dataset it is best possible to check if the EA is able to select the correct inputs and to cover the interesting regions by rules. The artificial dataset was created by generating 31 times 31 uniformly distributed data points of the function given by

$$g_2(u_1,u_2) = 8 \cdot \sin(10u^2 + 5u + 1) \cdot 2\left(e^{-(\frac{u_2-0.1}{0.25})^2} - 0.8 \cdot e^{-(\frac{u_2+0.75}{0.15})^2} - 0.4 \cdot e^{-(\frac{u_2-0.8}{0.1})^2}\right).$$

The model output is illustrated in Fig. 6.1 and the function name g_2 is taken over from the below mentioned papers. Originally the problem is a function approximation problem [128] and it was shown that this can be done very efficiently using a b-spline based model [196]. The authors used a complete rulebase and presented the function approximator only two features and the concerning output. A very high accuracy was obtained by using only 60 receptive fields (rules).

In the following the original input space will be widened by 20 additional inputs, which are noisy versions of the original ones. The noise ratio reaches from standard deviation 0.05 to standard deviation 0.5 with mean zero of the original values. Thus,

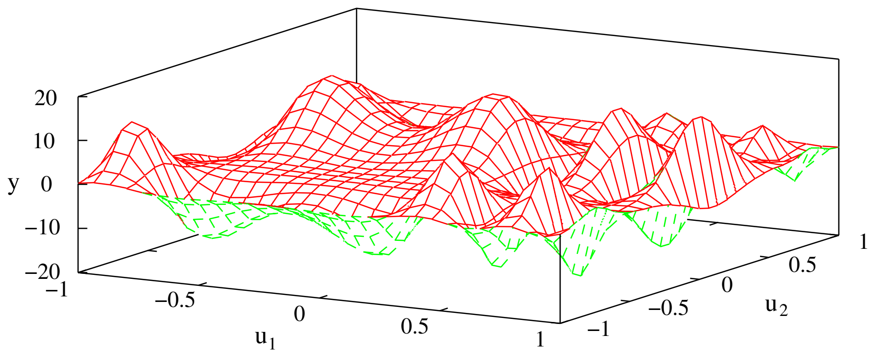


Figure 6.1: The artificial dataset.

the task is now to do a function approximation and to select the best possible input feature set to perform this function approximation. Input columns 0 and 11 (see Fig.6.2) were originally used to calculate the desired output depicted in Fig. 6.1. We expect that the EA identifies these originally used inputs, or at least that the selected DTSFM utilizes features with a low noise ratio.

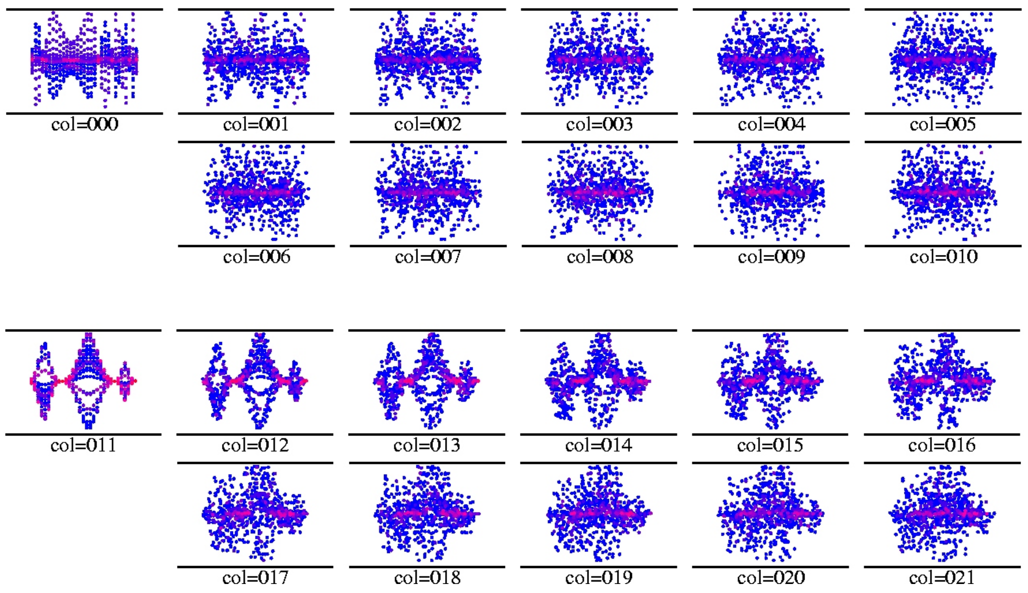


Figure 6.2: The used input dataset to approximate the function g_2. Column 1 to 10 are noisy versions of column 0 and column 12 to 21 are noisy versions of column 11. The input on the x-axis is plotted against the concerning output (of the noiseless input) on the y-axis.

Three evolutionary runs utilizing genotype-template 5.1 were performed. The EA was implemented as described in Sec. 5.4 beside that the fitness penalizing instructions (16)-(17) in Alg. 5.7 were not used. All three evolutionary modeling processes were activated with identical parameters and a different random generator initialization. The best run results in a model which produces a mean square error (MSE) of 10.1345 by using 49 rules. The MSE was calculated by presenting the identified model the same input patterns as in [128, 196]. The used inputs and the coverage with fuzzy sets of this model are depicted in Fig 6.3 and the concerning model output is shown in Fig. 6.4(b).

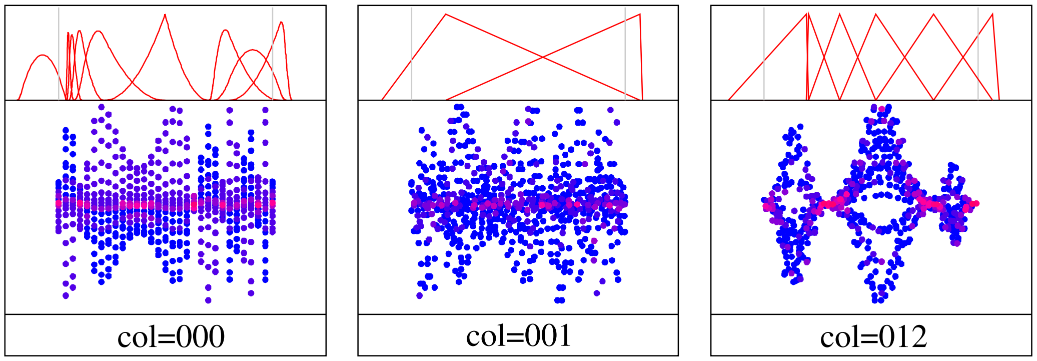
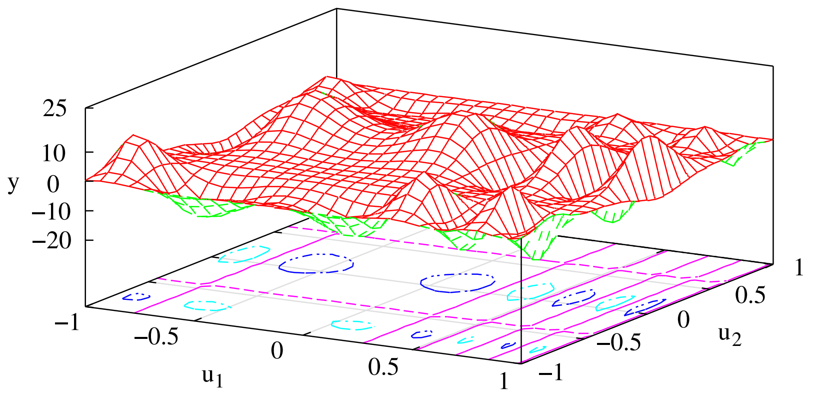


Figure 6.3: Used fuzzy sets to approximate function output depicted in Fig. 6.4(b).

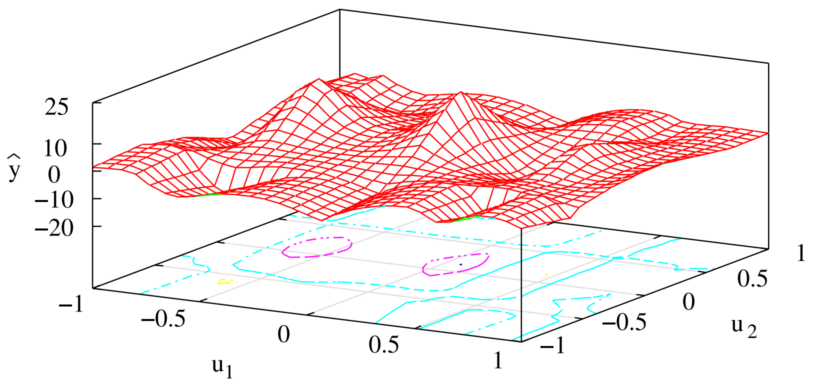
The mean MSE of all three runs result in 10.92 and the smallest model uses only 21 rules. In comparison to the results in [196] this is not so impressive (MSE = 2.8 with 60 rules), but by considering that an approach with equi-spaced b-spline distribution and a complete rulebase with 64 rules (without ”don’t care” premises) leads to a MSE of 10.91 the results are more than acceptable.

Furthermore, it has alway to keep in mind that the EA also has to select the relevant inputs and to choose premises from a huge amount of possible premises. Because of this a cross-validation size of 31 was used. Other parameters were set to one hour calculation time (AMD XP2000+ CPU) leading to, depending on the run, 59 to 85 generations[1]. Each generation a population of 80 individuals were calculated.

[1]It should be remarked that the algorithms were not fully optimized for speed. For example lookup tables could be used more extensively to speed up the calculations. The author estimates that a two to ten times faster computation is possible.



(a) Original function output.



(b) Approximated function output.

Figure 6.4: Original and approximated function output.

As parameters for the target model *min/max premise* = 1/3, *min/max order* = 1/3, *min/max coded rule* = 10/100, *min/max kernel* = 1/15 and *min/max feature* = 1/3 were chosen. Note that the three best models use exactly two premises in each rule.

The same parameters were used to calculate three ”headless chicken” [143] test runs by simply omitting crossover in the EA. All of these runs show slower convergence behavior and all runs lead to higher model MSEs as obtained by the worst identified model using crossover. After these encouraging results with artificial data, the approach has to confirm the shown capabilities by applying it to a complex real world problem.

6.2 Real World Data

In February 2004 approximately 23 million [83] chemical compounds were registered in the Chemical Abstract Service. Because of their abundance and wide use in numerous fields of production, a better understanding of their ecotoxicological impact on plant life, wild life, and the environment in general is of high interest. Apart from the ethical considerations associated with the use of animals, models which can give a clue to toxicity are of highly economical use, because they avoid useless, time consuming and expensive pilot batches.

6.2.1 Quantitative Structure Activity Relationships

In the following the widely accepted assumption is supported that macroscopic properties like toxicity and ecotoxicity strongly depend on microscopic features and the structure or the similarity [159] of molecules. This assumption is referred as *quantitative structure activity relationships* and was applied in the past years to a wide variety of chemical, biological, physical, and technological properties [84, 114].

6.2.2 Data Description

The used toxicity dataset was built up by the U.S Environmental Protection Agency [44,45,46] by starting from a revision of experimental data from literature. The dataset is one of the biggest available and furthermore very reliable [162]. Nevertheless, the dataset is based on experimental results involving living beings and thus, the testbed could never be identical. Therefore, the dataset is more or less noisy with a certain probability of containing outliers. The used dataset contains 568 organic compounds commonly used in industrial processes. Each compound is described by 167 molecular descriptors (see Sec. 6.2.3) and one toxicity value. Twelve of the molecular descriptors (feature 126-137) provide no information because the minimum descriptor value equals the maximum descriptor value for all molecules. These descriptors were removed from the dataset yielding in a total number of 155 descriptors. For a deeper chemical inside the interested reader is referred to [96]. The dataset was used in the European Community project IMAGETOX [79] (Intelligent Modeling Algorithms for General Evaluation of TOXicities) and according to the project rules, it is not allowed to distribute the data. However, people can easily ask the project member Dr. Emilio Benfenati [11] to obtain it.

The toxicity value referres to the acute toxicity for the fish species fathead minnow (Pimephales promelas) and is expressed as -log(LC50(mmol/l)), with LC50 as abbreviation for lethal concentration with 50% death rate after 96 hours. Thus, a high toxicity value expressed in the -log(LC50(mmol/l) measure is induced by the fact that only few molecules per mmol are needed to cause the above mentioned death rate. A high -log(LC50(mmol/l) value stands for a high aquatic toxicity.

6.2.3 Molecular Descriptors

The descriptors are used to mathematically characterize the molecules. Many of the descriptors were calculated by the Environmental Chemistry and Toxicology Laboratory at Istituto Mario Negri [118], using special software like Hyperchem 5.0, CODESSA 2.2.1 and Pallas 2.1. The set of descriptors can be split, according to the classification schema present in CODESSA [96] into six categories.

Constitutional descriptors depending on the number and type of atoms, bonds, and functional groups.

Geometrical descriptors contain information about the molecular surface area and volume, moments of inertia, shadow area, projections, and gravitational indices.

Topological descriptors are molecular connectivity indices which are related the the degree of branching in the compounds.

Electrostatic descriptors such as partial atomic charges and other depending on the possibility for some sites in the molecule to form hydrogen bonds.

Quantum-Chemicals descriptors like the total energy of the molecule, ionization potentials, the energies of the lowest unoccupied and highest occupied orbital, etc.

Physicochemical descriptors such as logD pH5.

In the chemical community it is common to classify descriptors with respect to their correlation to the desired output y. The Pearson Product-Moment Correlation Coefficient or correlation coefficient for short is in the following referred as R. R is a measure of the degree of linear relationship between two variables, i.e. the experimental toxicity value y and the by a model computed toxicity value $\hat{y}$. While in regression the emphasis is on predicting one variable from the other, in correlation the emphasis is on the degree to which a model may describe the relationship between two variables. In regression the interest is directional, one variable is predicted and the other is the predictor; in correlation the interest is non-directional, the relationship is the critical aspect. The correlation coefficient R may take on any value between plus and minus one, where the sign of the correlation coefficient defines the direction of the relationship, either positive or negative. In this thesis this direction is not important and therefore only absolute correlations values are taken into account.

Many papers use instead of R the squared correlation value to present the achieved results. In the following also squared correlation values, referred as R^2 will be used, where R^2 is given by

$$R^2 = \left(\frac{\sum_{m=1}^{M} (y_m - \bar{y})(\hat{y}_m - \bar{\hat{y}})}{\sqrt{\sum_{m=1}^{M} (y_m - \bar{y})^2} \sqrt{\sum_{m=1}^{M} (\hat{y}_m - \bar{\hat{y}})^2}} \right)^2, \tag{6.1}$$

with M : number of experimental toxicity measurements,

$$\bar{y} = \frac{1}{M} \sum_{m=1}^{M} y_m,$$

$$\bar{\hat{y}} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}_m.$$

The squared correlation is also known as the *coefficient of determination*. It is one of the best means for evaluating the strength of a relationship. For example, we know that the correlation between experimental toxicity and predicted toxicity is $R = 0.8$. If we square this number we will find $R^2 = 0.64$. Thus, 64 percent of the experimental toxicity is directly accounted for the predicted toxicity and vice versa. For fast comparison Tab. 6.1 list the descriptor classification and the concerning correlation and squared correlation values.

Classification	Correlation	Squared Correlation
substantial descriptors	$\mid R \mid \geq 0.99$	$\mid R^2 \mid \geq 0.9801$
important descriptors	$0.99 > \mid R \mid \geq 0.80$	$0.9801 > \mid R^2 \mid \geq 0.64$
likely descriptors	$0.80 > \mid R \mid \geq 0.50$	$0.64 > \mid R^2 \mid \geq 0.25$
specific descriptors	$0.50 > \mid R \mid$	$0.25 > \mid R^2 \mid$

Table 6.1: Common classification of molecular descriptors.

6.2.4 Toxicity Prediction With Multi-Linear Regression

A simple (multi)linear regression should always be one of the first steps in modeling a new dataset. Therefore, the best descriptor concerning the correlation to the output was computed by a linear regression. Furthermore the best possible combination of up to four descriptors was calculated by computing the resulting correlations of all possible permutations. The best four-dimensional linear model results in a squared correlation to the experimental obtained output of $R^2 = 0.6482$. Tab. 6.2 lists the results and the identified (multi)linear models to obtain these results. All more flexible models have to yield a better accuracy to legitimate them-self.

R^2	Feat.	Feature Name	Polynomial Model
0.4773	151	logD pH5	$0.0356386 + 0.447185 \cdot u_0$
0.6056	41	Molecular weight	$-1.21074 + 0.00884406 \cdot u_0$
	155	logD pH9	$+0.340867 \cdot u_1$
0.6269	11	Relative number of H atoms	$0.0593391 - 2.3744 \cdot u_0$
	50	Kier & Hall index (order 0)	$+0.196906 \cdot u_1$
	155	logD pH9	$+0.334053 \cdot u_2$
0.6482	49	Randic index (order 3)	$-1.32979 + 0.228086 \cdot u_0$
	53	Kier & Hall index (order 3)	$+0.286063 \cdot u_1$
	99	Topographic electronic index	$-0.834284 \cdot u_2$
	155	logD pH9	$+0.235826 \cdot u_3$

Table 6.2: Best found squared correlation values by performing (multi)linear regressions for all possible models using one, two, three and four input features.

6.2.5 Toxicity Prediction With DTSFMs

All together 90 evolutionary runs were performed to calculate a validated model for toxicity prediction. All of these runs were done by utilizing genotype-template 5.1. Again the EA implementation as described in Sec. 5.4 was applied. The runs can be divided into three experiment. In the first and second experiment, each involving 36 EA runs, the complete available descriptor set was used as input for the model identification process. In these experiments no interpretability measure was used to modify the fitness of the candidate models. Parameters differ only in the number of allowed input features, allowed number of premises of a rule and the cross-validation size.

The most often selected features of all 72 computed models are than used in Sec. 6.2.5.3 to build a reduced descriptor set. On this reduced descriptor set again 18 models are calculated, this time with a fitness penalized by the fulfillment value of an interpretability factor.

6.2.5.1 First Experiment Allowing Three Premises

In the evolutionary runs of the first experiment the maximal allowed premises were set to three. This was done by setting the genotype-template variables *min/max premises = 1/3*. All together 36 runs were calculated with three different settings for the genotype-template variables *min/max features*, namely 1/5, 1/10 and 1/15. The evolutionary parameters for each of the runs were set to population size = 100, tournament size = 10 and one hour calculation time was chosen as stop criterion. Furthermore one half of the runs used as fitness for the candidate models a 8 time cross-validated normalized error estimation and the other half a 71 time cross-validated normalized error estimation. The chosen cross-validation sizes are founded on the available number of 568 molecules. Thus, a cross-validation size of 8 yields in data-subsets of size 71 and vice versa.

For each different parameter setting six runs were performed. Table 6.3 shows the best, the mean and the worse model results for each setting. The DTSFM resulting in the bold red printed values is presented in greater detail.

The inputs of this model are listed in Tab. 6.4. Figure 6.5 shows the used fuzzy sets and the concerning data distribution of the model inputs. The calculated model output is plotted against the experimental toxicity values, which is illustrated in Fig. 6.6.

		best			mean		worst	
max Feat.	CV	NMSE	R^2	Rules	NMSE	R^2	NMSE	R^2
5	8	0.2304	0.6854	9	0.2403	0.6721	0.2485	0.6608
5	71	0.2317	0.6837	11	0.2451	0.6653	0.2601	0.6448
10	8	0.2220	0.6941	14	0.2308	0.6849	0.2476	0.6619
10	71	0.2219	0.6970	11	0.2315	0.6839	0.2457	0.6645
15	8	0.2166	0.7043	16	0.2197	0.7000	0.2228	0.6958
15	71	**0.2156**	**0.7056**	**12**	0.2266	0.6907	0.2327	0.6823

Table 6.3: Toxicity modeling results of experiment one. The model (bold-red) inputs, fuzzy sets and output are shown by Tab. 6.4, respectively, Fig. 6.5 and Fig. 6.6.

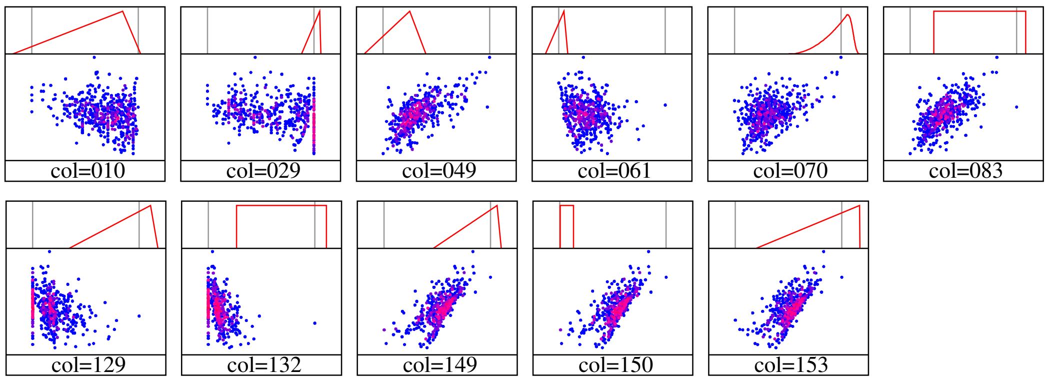


Figure 6.5: Input features as listed in Tab. 6.4 covered by the fuzzy sets with input-data (x-axis) versus experimental toxicity data (y-axis).

Descriptor Class	Feat.	Descriptor Name
Constitutional	10	Relative number of H atoms
Constitutional	29	Relative number of single bonds
Topological	49	Kier&Hall index (order 0)
Topological	61	Average Bonding Info. content (order 0)
Topological	70	Bonding Info. content (order 1)
Geometrical	83	ZX Shadow
Electrostatic	129	HA dependent HDSA-1
Electrostatic	132	HA dependent HDSA-2/TMSA
Physicochemical	149	logD pH3
Physicochemical	150	logD pH5
Physicochemical	153	logD pH7.4

Table 6.4: Used descriptors of the best model of Tab. 6.3.

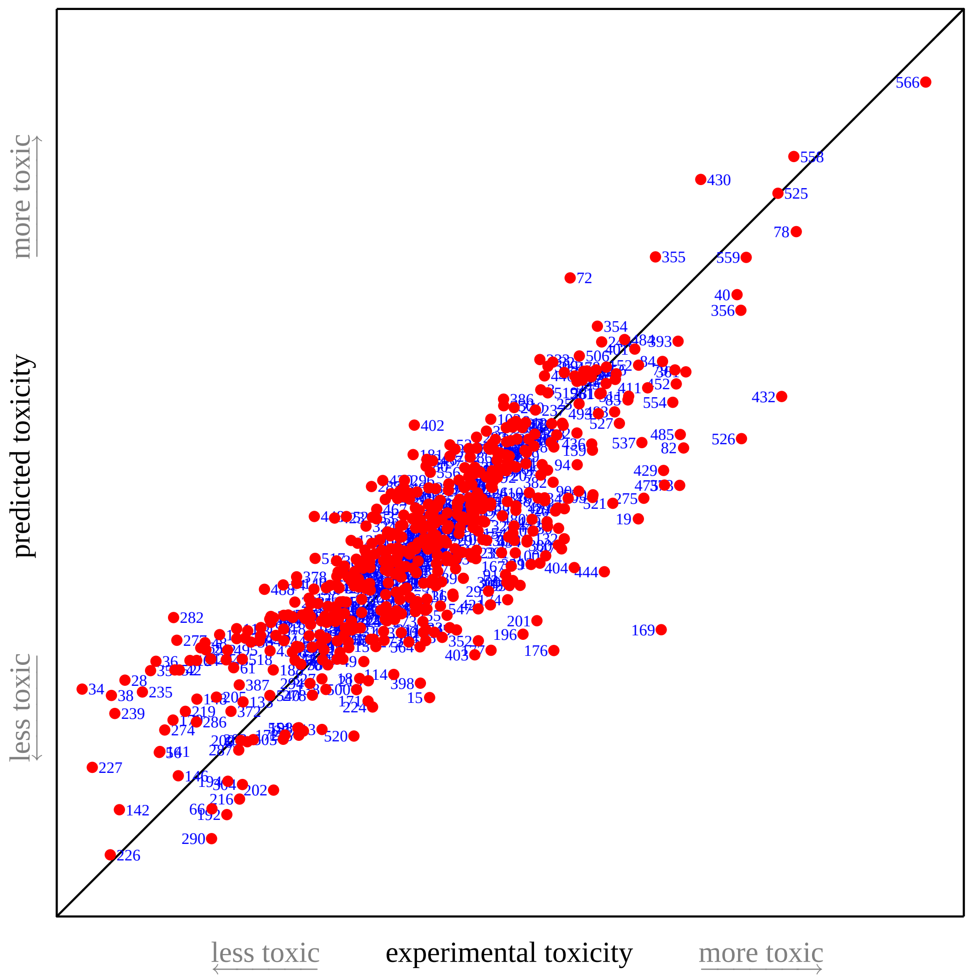


Figure 6.6: Experimental toxicity versus predicted toxicity of the best DTSFM of experiment one with model inputs as given in Tab. 6.4.

The illustrated model uses only 12 rules with a total of 14 premises. Thus, only two rules uses two premises. The squared correlation value is 0.7056 which is equivalent to a correlation of 0.84. A descriptor with this magnitude of correlation is referred as important and so should the model. It is interesting that very few of the calculated models make use of more than one premise in each rule. This could be caused by an inadequacy search of the EA or because of easy to model QSAR. The latter is more likely since the complex relationships of the artificial dataset were established by the EA. On the other hand it is not possible to model to many special dependencies because a high number of dependencies need also a high number of data pattern to model the process. To check if more premises will be used if more premises are allowed in the modeling process experiment two was performed.

6.2.5.2 Second Experiment Allowing Several Premises

The second experiment is a retake of the first one, except that more premises are allowed in the evolutionary search. To be precisely in this experiment the number of allowed premises is set to the number of allowed features. Again 36 models (always six with the same settings) were calculated and the results are listed in Tab. 6.5. Because of the bigger search space and the possible high flexibility of the candidate models in the following the most flexible model with the more reliable cross-validation size is depicted in greater detail, although it is not the best found model.

The computed squared correlation of this model to the experimental toxicity values is 0.6784. The corresponding correlation value is 0.8237 which classifies this model also as important. The model uses 10 rules, each rule consisting of only one premise. The overall convergence of the EA is slower than in the first experiment and it can be seen that for 10 and 15 features the 8 time cross-validated models clearly outperform the 71 times cross-validated ones. In general this is an indication for overfitting, but if so, the concerning models should offer a higher flexibility as the 71 times cross-validated ones. This is not the case since nearly all models use one premise in each rule. The conclusion is the same as in experiment one. It seems that there exist simple and substantial relations between the descriptor values of a molecule to the toxicity of this molecule. It is most likely that a modeling of more specific dependencies needs more data and/or more precise experimental toxicity values.

But there is no reason to dramatize. In comparison to other QSAR modeling methods [123, 124] using the same dataset, the presented results seems to be superior in prediction accuracy as well as in model simplicity (and therefore more reliable), although a direct comparison is not really possible due to the different validation procedures. To further increase the accuracy of the modeling process, the input dataset is reduced to decrease the search space for candidate models. This is done by selecting the most often used features in models of this and the first experiment as listed in Tab. 6.7. For the interested reader the never used features are also listed in Tab. 6.8.

		best			mean		worst	
max Feat.	CV	NMSE	R^2	Rules	NMSE	R^2	NMSE	R^2
5	8	0.2303	0.6856	9	0.2456	0.6647	0.2627	0.6413
5	71	0.2336	0.6813	7	0.2477	0.6620	0.2634	0.6410
10	8	0.2143	0.7074	13	0.2356	0.6784	0.2528	0.6548
10	71	0.2369	0.6765	8	0.2437	0.6672	0.2551	0.6517
15	8	0.2182	0.7020	15	0.2383	0.6748	0.2584	0.6474
15	71	**0.2356**	**0.6784**	**10**	0.2504	0.6581	0.2774	0.6212

Table 6.5: Toxicity modeling results of experiment two. The model (bold-red) inputs, fuzzy sets and output are shown by Tab. 6.6, respectively, Fig. 6.7 and Fig. 6.8.

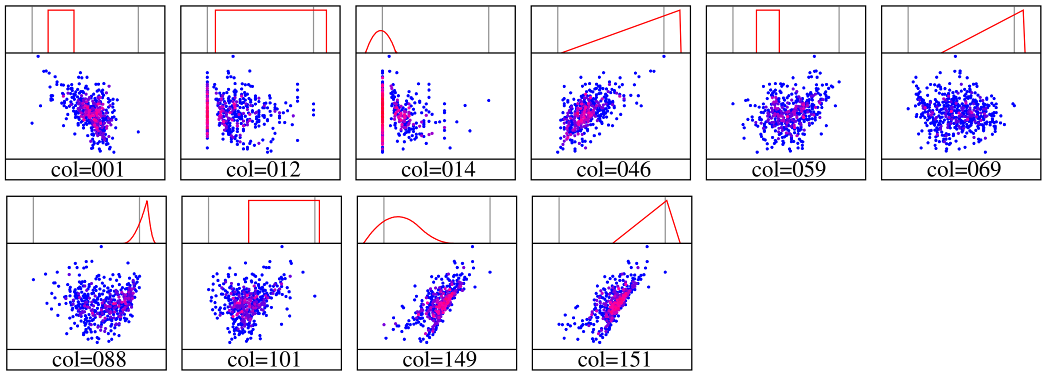


Figure 6.7: Input features as listed in Tab. 6.6 covered by the fuzzy sets with input-data (x-axis) versus experimental toxicity data (y-axis).

Descriptor Class	Feat.	Descriptor Name
Quantum-Chemicals	1	Binding Energy (kcal/mol)
Constitutional	12	Relative number of O atoms
Constitutional	14	Relative number of N atoms
Topological	46	Randic index (order 1)
Topological	59	Average Complementary Info. content (order 0)
Topological	69	Average Bonding Info. content (order 1)
Geometrical	88	ZX Shadow / ZX Rectangle
Electrostatic	101	PPSA-1 Partial positive surface area
Physicochemical	149	logD pH3
Physicochemical	151	logD pH6.5

Table 6.6: Used descriptors of the best model of Tab. 6.5.

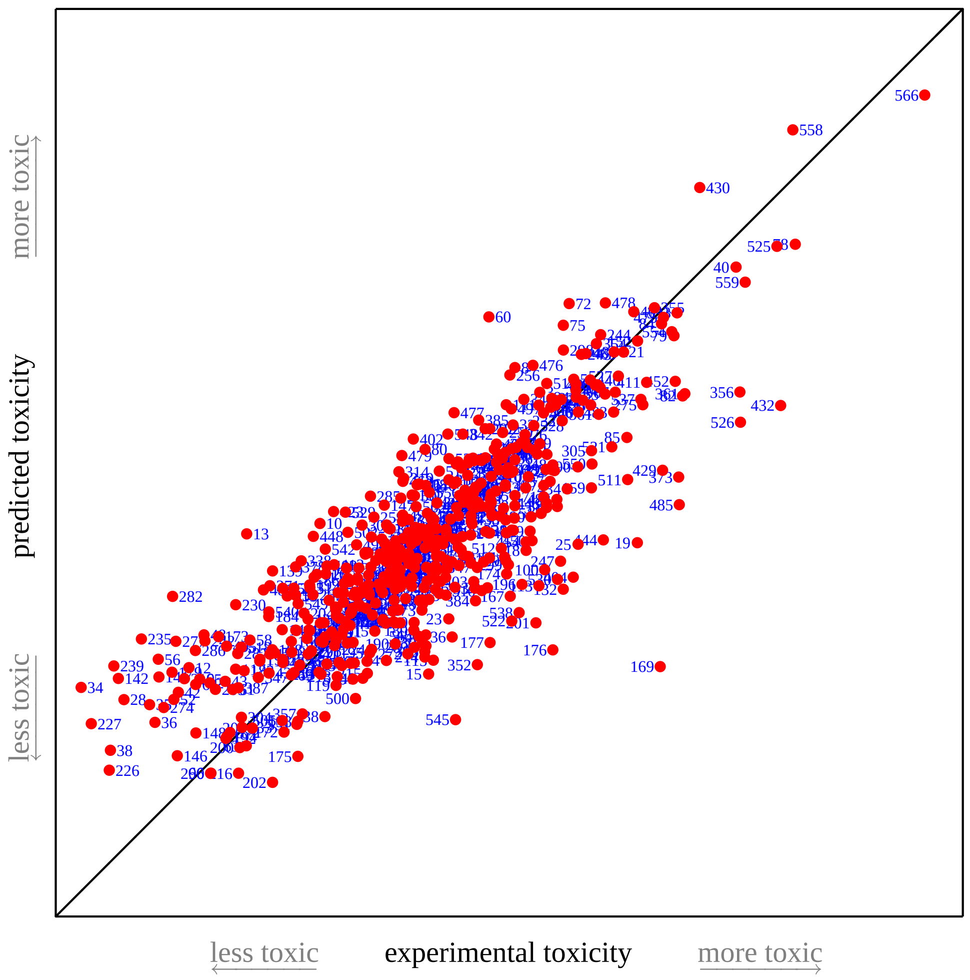


Figure 6.8: Experimental toxicity versus predicted toxicity of the selected DTSFM of experiment two with model inputs as given in Tab. 6.6.

Index	Descriptor Name	Descriptor Class	Frequency
2	Heat of Formation (kcal/mol)*	Quantum-Chemicals	5
5	LUMO (eV)*	Quantum-Chemicals	16
8	Relative number of C atoms	Constitutional	9
10	Relative number of H atoms	Constitutional	16
12	Relative number of O atoms	Constitutional	12
29	Relative number of single bonds*	Constitutional	16
40	Molecular weight*	Constitutional	13
42	Gravitation index (all bonds)	Constitutional	11
43	Gravitation index (all pairs)	Constitutional	6
46	Randic index (order 1)	Topological	6
48	Randic index (order 3)	Topological	8
49	Kier&Hall index (order 0)*	Topological	14
53	Kier shape index (order 1)	Topological	9
64	Information content (order 1)	Topological	11
65	Average Structural Information content (order 1)	Topological	5
66	Structural Information content (order 1)	Topological	9
69	Average Bonding Information content (order 1)	Topological	6
70	Bonding Information content (order 1)	Topological	8
74	Structural Information content (order 2)	Topological	6
77	Average Bonding Information content (order 2)	Topological	9
80	Moment of inertia A	Geometrical	5
83	XY Shadow	Geometrical	5
88	ZX Shadow / ZX Rectangle	Geometrical	18
90	Molecular volume / XYZ Box	Geometrical	8
95	Min partial charge (Qmin)	Electrostatic	6
104	FPSA-1 Fractional PPSA (PPSA-1/TMSA)	Electrostatic	6
105	FNSA-1 Fractional PNSA (PNSA-1/TMSA)	Electrostatic	8
115	PPSA-3 Atomic charge weighted PPSA*	Electrostatic	6
116	PNSA-3 Atomic charge weighted PNSA	Electrostatic	5
118	FPSA-3 Fractional PPSA (PPSA-3/TMSA)*	Electrostatic	20
119	FNSA-3 Fractional PNSA (PNSA-3/TMSA)	Electrostatic	5
149	logD pH3	Physicochemical	18
150	logD pH5	Physicochemical	16
151	logD pH6.5	Physicochemical	11
152	logD pH7	Physicochemical	6
153	logD pH7.4	Physicochemical	18
154	logD pH9*	Physicochemical	19

Table 6.7: Most frequently used molecular descriptors in all evolutionary computed DTSFMs. The eight descriptors marked with a star were also used in [122] who selected 17 descriptors (16 of the descriptors are present in the dataset used for this thesis) out of a nearly identical descriptor-set with help of a principle component analysis.

Index	Descriptor Name	Descriptor Class
16	Relative number of S atoms	Constitutional
17	Number of F atoms	Constitutional
18	Relative number of F atoms	Constitutional
20	Relative number of Cl atoms	Constitutional
23	Number of I atoms	Constitutional
24	Relative number of I atoms	Constitutional
26	Relative number of P atoms	Constitutional
27	Number of bonds	Constitutional
30	Number of double bonds	Constitutional
31	Relative number of double bonds	Constitutional
56	Info. content (order 0)	Topological
60	Complementary Info. content (order 0)	Topological
62	Bonding Info. content (order 0)	Topological
73	Average Structural Info. content (order 2)	Topological
81	Moment of inertia B*	Geometrical
82	Moment of inertia C	Geometrical
84	XY Shadow / XY Rectangle	Geometrical
85	YZ Shadow	Geometrical
86	YZ Shadow / YZ Rectangle	Geometrical
99	Topographic electronic index (all bonds)	Electrostatic
109	PNSA-2 Total charge weighted PNSA	Electrostatic
110	DPSA-2 Difference in CPSAs (PPSA2-PNSA2)	Electrostatic
112	FNSA-2 Fractional PNSA (PNSA-2/TMSA)	Electrostatic
113	WPSA-2 Weighted PPSA (PPSA2*TMSA/1000)	Electrostatic
122	RPCG Relative positive charge (QMPOS/QTPLUS)	Electrostatic
123	RPCS Relative positive charged SA (SAMPOS*RPCG)	Electrostatic
130	HA dependent HDSA-1/TMSA	Electrostatic
133	HA dependent HDSA-2/SQRT(TMSA)	Electrostatic
135	HA dependent HDCA-1/TMSA	Electrostatic
142	HASA-2/TMSA	Electrostatic
145	HACA-1/TMSA	Electrostatic
146	HACA-2	Electrostatic
148	HACA-2/SQRT(TMSA)	Electrostatic

Table 6.8: Molecular descriptors never used in all evolutionary computed DTSFMs. It is notable that one of the descriptors, namely ”Moment of inertia B” was one of 17 selected descriptors in [122].

6.2.5.3 Model Identification on the Reduced Feature Set

As mentioned above the third experiment was performed using a reduced feature set with molecular descriptors as given in Tab. 6.7. This feature set was derived by using all features which were used at least five times during experiment one and experiment two. Figure 6.9 gives an overview of the distribution of used features during the modeling process in experiment one and two. In the following evolutionary runs only 37 inputs were considered, which significantly reduces the search space for an optimal model.

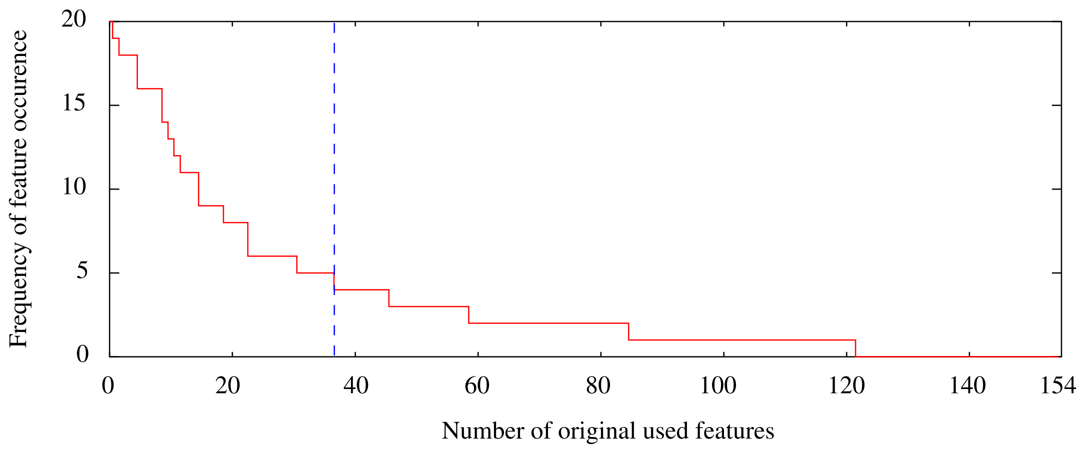


Figure 6.9: Frequency distribution of used features in evolutionary identified models using the complete feature set. The dashed blue line indicates that for the final model only the 37 most often selected features were used.

With help of this experiment the final best model should be identified. Because of this it becomes necessary to make use of the implemented penalizing scheme, which favorites interpretable models. All parameters were chosen as in experiment one, except that only the more reliable 71 times cross-validation procedure is used and that the fitness of each candidate DTSFM is proportional to the normalized MSE and the penalty factor computed by Eq. (3.13). This penalizing scheme should encourage the modeling process to use models with at most four simultaneously activated rules.

The summarized results of the 18 evolutionary runs are presented in Tab. 6.9. Although the best model yield in a squared correlation of 0.7021 to the experimental toxicity the slightly worse, but much smaller model is presented in greater detail. This very sparse model uses only six rules with eight premises and its computed output has a squared correlation to the experimental toxicity values of 0.7002 (normal correlation is 0.8368). Table 6.10 lists the features selected by this model and Fig. 6.11 illustrates the computed versus the experimental toxicity values. The interpretability value of simultaneously activated rules were for all final models calculated to one, meaning that in all models at most four rules are activated simultaneously. Thus, the presented model is fully interpretable in terms of the very strict objective IM defined in Sec. 3.3.3.

		best			mean		worst	
max Feat.	CV	NMSE	R^2	Rules	NMSE	R^2	NMSE	R^2
5	71	**0.2196**	**0.7002**	**6**	0.2428	0.6685	0.2629	0.6410
10	71	0.2252	0.6925	7	0.2321	0.6830	0.2384	0.6744
15	71	0.2182	0.7021	10	0.2284	0.6881	0.2385	0.6744

Table 6.9: Toxicity modeling results with the reduced feature set. The model (bold-red) inputs, fuzzy sets and output are shown by Tab. 6.10, respectively, Fig. 6.10 and Fig. 6.11.

Descriptor Class	Feat.	Descriptor Name
Constitutional	3 (10)	Relative number of H atoms
Constitutional	4 (12)	Relative number of O atoms
Topological	11 (49)	Kier&Hall index (order 0)
Geometrical	21 (88)	ZX Shadow / ZX Rectangle
Physicochemical	36 (154)	logD pH9

Table 6.10: Used descriptors of the highlighted model of Tab. 6.9.

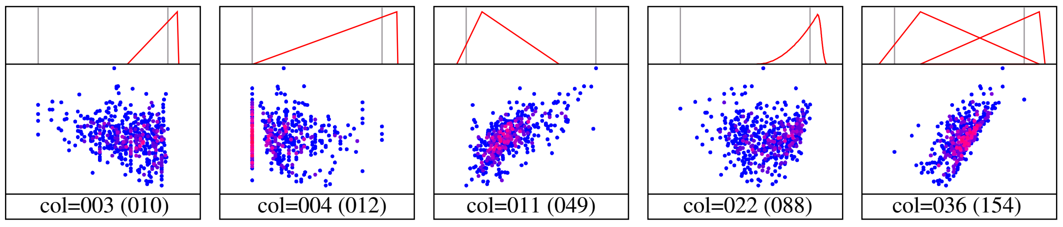


Figure 6.10: Input features (see also Tab. 6.10) covered by the fuzzy sets used in the DTSFM found on the reduced feature set with the concerning input (x-axis) versus toxicity (y-axis) data. For the FM output see Fig. 6.11.

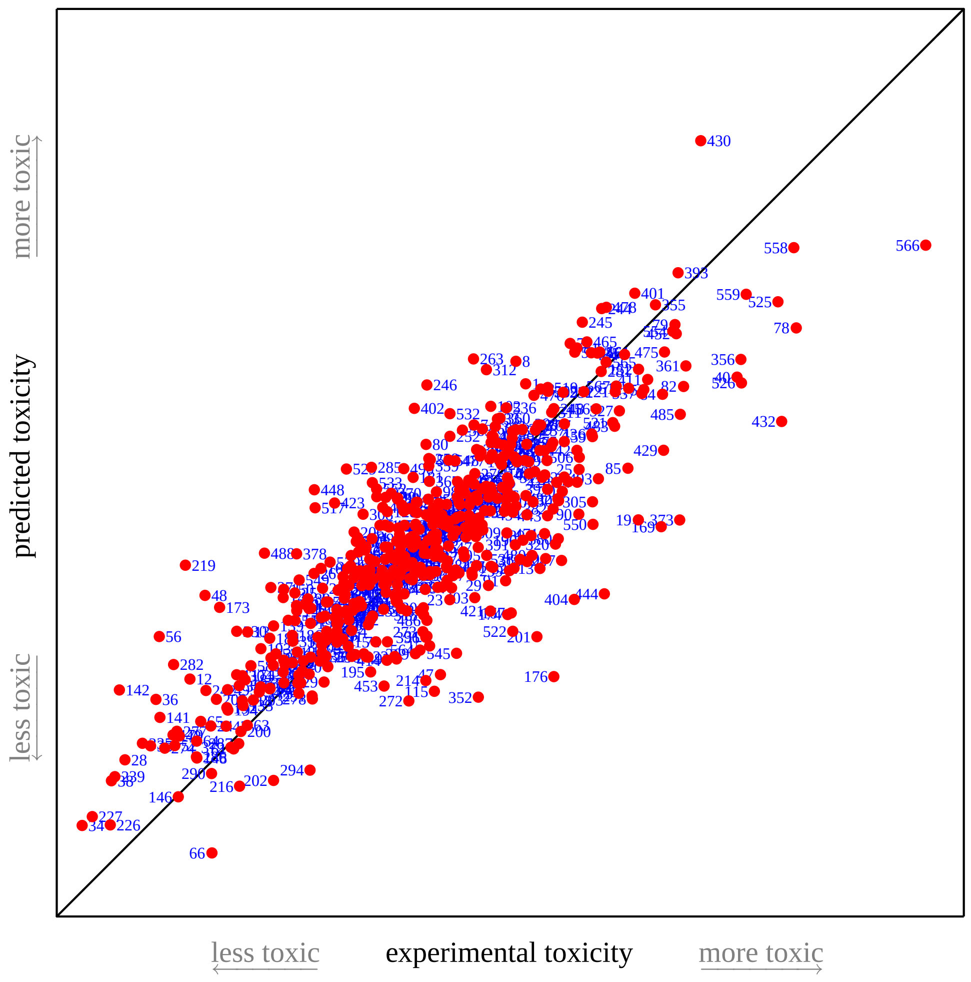


Figure 6.11: Experimental toxicity versus predicted toxicity of the DTSFM found on the reduced feature set (see Tab. 6.9). The used model inputs are given in Tab. 6.10.

6.3 Summary

In this chapter the described methods and algorithms were successfully tested on an artificial and, more important, on a complex real world dataset. It was shown that tree-based genotypes, which are defined with the help of grammar-based genotype-templates, are capable to perform complex system identification tasks. In example important features were selected out of a feature set consisting of 155 molecular descriptors. These features were covered by fuzzy sets and the resulting model is, with respect to the IM measure defined in Sec. 3.3.3, fully interpretable. Thus, not only the functionality of the system identification was demonstrated, but also interesting information for scientists who work in the field of QSAR was provided. The next chapter will shortly summarize this thesis and will also give a selective view to possible future work.

Chapter 7

Conclusions

This chapter will briefly summarize the proposed and developed concepts and their application presented in this thesis. Furthermore some possible extensions are proposed.

7.1 Brief Summary of Work and Discussion

Firstly a motivation for and an introduction to system identification was given and by doing so, the crucial problems which occur during data-driven modeling were discussed. The tasks in each system identification process were listed as selecting an analytical expression as framework for the model, selecting the model structure and to perform parameter optimization of the model. The mathematical problem of parameter optimization in overdetermined systems were presented in greater detail. It was justified that complexity and flexibility of a model should not be confused. Based on this the common generalization error estimation methods were listed and observed which of them are applicable for pure data-driven modeling.

The introspection of system identification was followed by the decision to use the class of zero order Takagi-Sugeno fuzzy models as analytical expression. This class of models were chosen because of their possibility to utilize powerful learning algorithms based on direct least squares and because of the fact that the resulting models can be refined or analyzed by human experts, if necessary. To retain this accessibility for human beings, interpretability factors concerning the rule-base of Takagi-Sugeno fuzzy models were formulated and assembled to an objective interpretability measure. The proposed interpretability factors are easy to calculate and thus, the resulting interpretability measure might be beneficial for the comparisons of models derived by different model identification schemes. Thanks to the ”good” choice of using b-splines as fuzzy sets not all interpretability factors had to be implemented, because some were inherently fulfilled. Nevertheless, a provision of such interpretability factors is indispensable since models which are claimed to be ”interpretable” can not be compared only on the base of accuracy.

After this the model structure selection method was chosen to base on evolutionary computation. This decision was justified by the fact that evolutionary methods are, at least theoretically, capable to provide an optimal solution for the structure identification problem. A detailed insight to evolutionary computation and especially to the problems related with an optimal genotype representation were given.

On the base of these insights a novel and grammar based method was proposed to formulate arbitrary genotype-templates. These genotype-templates provide a general concept to define genotype search spaces, which cope with the observation that in EC the challenge shifts more and more from finding a solution for the original problem, to the task of finding a possible and adequate genotype encoding.

Moreover, a tree based genotype representation was favored and the evolutionary operations for this kind of representation were described and fully implemented. The implementation was described in greater detail and the most important algorithms were given in pseudo-code. With help of tree-based genotype representations, which are instances of a search space defined by the concerning genotype-templates, it becomes possible to build commonly usable libraries, comprising genotype-templates and the concerning genotype to phenotype mapping functions. Thus, this thesis gives a possible solution to avoid unnecessary reimplementations of EAs. By using the proposed method the concepts of modularity and reusability are applied to the design of EAs. The selected task of system identification via DTSFMs provides a valuable problem which was described, implemented and tested. Furthermore all relevant topics which are possibly of interest for Takagi-Sugeno fuzzy modeling were outlined and often discussed in greater detail.

The capabilities of the method was demonstrated on an artificial dataset and a complex real world system. The first was done for testing and the second to perform a challenging and meaningful task. During the modeling experiments important descriptors for toxicity prediction were identified and a sparse and accurate model was obtained.

7.1.1 Inherently Fulfilling the Leveling Interpretability Condition

As already mentioned the so called leveling interpretability condition for fuzzy sets was not implemented for the used experiments. This is firstly caused by the fact that the artificial dataset was only motivated to test the implemented algorithms without further intention to interpret the found models. Concerning the real world problem the problem of different levels of maximum activations of fuzzy sets never emerged (at least not in the best found models), because features covered by fuzzy sets of order three b-splines were containing only one active fuzzy set.

The second and more relevant reason of the non-implementation of the leveling condition is caused by the insight that the b-spline approach can be extended in such a way that fuzzy sets based on this extension inherently fulfill the leveling condition. This can be done without loosing the flexibility of b-spline based methods. In [157] it was suggested that a fuzzy set could be constructed by more than one b-spline. This was motivated by the attempt to overcome the direct relationship between the width of the univariate basis functions and its order. As already mentioned in [18] it might be necessary to have wide basis functions, for instance, to increase the initial rate of convergence. In [18] this was implemented by so called dilated basis functions. The extension proposed here to the standard b-spline approach is similar but not equivalent to the use of dilated b-splines.

An extension fulfilling the leveling condition is based on a combination of at least $order - 1$ b-splines defined over a knot-vector consisting of $order \cdot 2 - 1$ knots. By doing so the maximum accumulated activation value of such combined b-splines is guaranteed to be one.

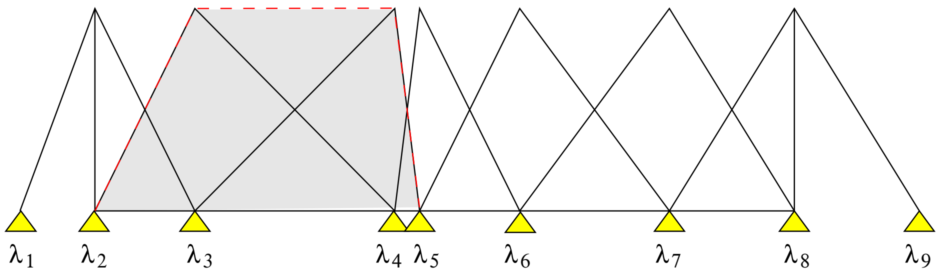


Figure 7.1: Combined b-splines forming a more capable fuzzy set.

Furthermore, by combining a number of b-splines equal to their order, utilizing for each combined b-spline a knot-vector of $order \cdot 2$ knots, the restriction of b-spline based fuzzy sets (order > 1), which have only a single point as maximum activation support, vanishes (see Fig. 7.1).

A careful implementation of combined b-splines will result in first class fuzzy sets, because they inherently fulfill the stated complementarity and leveling interpretability factors.

7.2 Future Work

This section provides some thoughts which might be promising research areas.

7.2.1 Incorporation of Process Knowledge

As mentioned in Sec. 5.4.5.3 prior available knowledge based on human expertise can already be considered in the formulation of the genotype-template and in the formulation of genotype constraints. If expert knowledge is available much of these knowledge concerns more general qualities of the output behavior of a system. For example ”here we expect a sharp positive peak” or ”in this area the output is near zero”. It is quite difficult to construct a hypothesis space already mentioning this kind of constraints. A more tractable way of considering these constraints is by penalizing the fitness of the candidate solutions proportional to the constraint violation. This strategy was also followed by the implementation of the interpretability constraint IF_3. Unfortunately we have to keep in mind that the more constraints are implemented by using penalties, the more the problem becomes multi-objective. There exist methods to deal with multi-objectivities in EC [31], but normally a good balance, to be precisely the desired Pareto-optimal point, of the different objectives have to be provided by the model-designer. Some approaches try to provide several Pareto-optimal candidate solutions to a problem. But these methods are restricted to simple problems or few objectives, since the number of Pareto-optimal solutions increases exponentially with the number of objectives, i.e. with the number of constraints implemented by penalizing candidate models.

It has also to be considered that qualitative expertise, which is not supported by data, will not be mentioned by parameter optimization techniques based on LS. In these cases the parameters have to be encoded into the genotype and at a certain time of the identification process these parameters have to be decoupled from the embedded LS based parameter optimization technique. After decoupling the parameters from the LS optimization these parameters are also subject of evolutionary changes, which are influenced by penalizing the fitness due to violation of the qualitative expertise. A first implementation and some interesting results concerning this technique are presented in [82].

7.2.2 Hierarchical Modeling

More and more complex real world systems become subject to modeling approaches and completely data-driven approaches are tentatively used to model them. The application area of fuzzy rule based models has reached economic and ecological fields. Overall, the complexity of systems which are tackled by modeling rises dramatically, but the framework of all used models hardly ever uses hierarchical structures, although the underlying real world problems must be assumed to be of hierarchical nature. Thus, it becomes more and more indispensable to deal with hierarchical models.

There exist adaptive hierarchical approaches like ASMOD [97] and real hierarchical approaches like NetFAN [80, 81]. Why it is not common to use them? Firstly, the necessity is only given for more complex problems and many, still not considered, low-complex problems have the potential to be solved by non-hierarchical models. Secondly, the demands to model hierarchical systems are far more challenging and thus the availability of implemented tools tends to zero.

Using FMs is nowadays an accepted approach and widely used in industry and thus, the scientific focus should (and also will) move to the modeling of high-complexity problems. Implementing hierarchical models needs many pieces and yet these pieces are not easily available in a bundle. Furthermore, by now there is no commonly used set of artificial benchmark data describing different hierarchical systems. The author of this thesis thinks that hierarchical modeling is an issue of highest interest, but it is an issue where still some requirements are missing.

7.3 Summary

This last chapter gave a short summary of the presented concepts and the newly developed methods in this thesis. Although a comprehensive guideline for the data-driven identification of descriptive Takagi-Sugeno fuzzy models via evolutionary computation was constituted, the last section indicates that interesting extensions are imaginable and thus, I curiously look forward to them.

Appendix A

List of Molecules

This appendix lists the molecules contained in the toxicity dataset used in Sec. 6.2. The list of molecules subsumes the molecule number used in this thesis, the concerning molecule name and a molecule code which was assigned by the US Environmental Protection Agency. This code refers to different chemical classes (see Tab. A.1 on page 139), according to a classification defined by the US Environmental Protection Agency.

1. ”4-(hexyloxy)-m-anisaldehyde” 5
2. ”5-bromo-2-nitrovanillin” 5
3. ”p-chlorophenyl-o-nitrophenyl ether” 3.1
4. ”3’-chloro-o-formotoluidide” 10.4
5. ”di-n-butylisophthalate” 8.1
6. ”1.1-diphenyl-2-propyn-1-ol” 4.2
7. ”4.7-dithiadecane” 12.1
8. ”4.9-dithiadodecane” 12.1
9. ”2-chloroethyl-n-cyclohexyl carbamate” 21
10. ”phenobarbital” 23
11. ”2.4-dinitrophenol #9” 14
12. ”urethane” 21
13. ”salicylic acid na+ #2” 7
14. ”benzamide” 8.2
15. ”1.1-dimethylhydrazine” 11.1
16. ”pentobarbital” 23
17. ”amobarbital” 23
18. ”caffeine” 23.1
19. ”2-methyl-1.4-naphthoquinone” 6.2
20. ”2.3.4.6-tetrachlorophenol” 14.1
21. ”4-chloro-3-methyl phenol #1” 14.1
22. ”tolazoline hydrochloride” 15.5
23. ”amphetamine sulfate” 23.1
24. ”diethyl ether” 3
25. ”strychnine hemisulphate salt” 22
26. ”aniline #1” 10.3
27. ”carbaryl (sevin) #2” 21
28. ”ethanol” 4
29. ”nicotine sulfate #1” 23.1
30. ”2-hydroxybenzamide” 8.2
31. ”hexanal #2” 5
32. ”dicumarol” 8
33. ”p-phenoxybenzaldehyde” 5
34. ”methanol-rhodamine b” 4
35. ”2-propanol #1” 4
36. ”acetone #1” 6
37. ”chloroform” 2
38. ”methyl sulfoxide” 12.3
39. ”hexachloroethane #1” 2
40. ”2.2’-methylene bis(3.4.6-trichlorophenol)” 14.1
41. ”4’-aminopropiophenone” 6
42. ”1-propanol#1” 4
43. ”1-butanol” 4
44. ”1-pentanol” 4
45. ”benzene #2” 13
46. ”1.1.1-trichloroethane #1” 2
47. ”thiopental.sodium salt” 23
48. ”acetonitrile” 9
49. ”ethanal #1” 5
50. ”dichloromethane” 2
51. ”iodoform” 2

52. ”2-methyl-2-propanol” 4
53. ”2.2.2-trifluoroethanol” 4
54. ”3.3-dimethyl-2-butanone” 6
55. ”pentachloroethane” 2
56. ”5.5-dimethylhydantoin” 15.5
57. ”3-methyl-3-pentanol” 4
58. ”3-methyl-1-pentyn-3-ol” 4.2
59. ”1-ethynyl-cyclohexanol” 4.2
60. ”tris(2-butoxyethyl) phosphate” 19
61. ”2-methyl-1-propanol” 4
62. ”1.2-dichloropropane” 2
63. ”1.2-diaminopropane” 10
64. ”2-butanol” 4
65. ”2-butanone” 6
66. ”1-amino-2-propanol” 4
67. ”1.1.2-trichloroethane” 2
68. ”trichloroethylene” 2.1
69. ”methyl acetate #1” 8
70. ”1.1.2.2-tetrachloroethane” 2
71. ”b-ionone” 6
72. ”4.4’-isopropylidenebis(2.6-dichlorophenol)” 14.1
73. ”p-tert-pentylphenol” 14
74. ”1.8-diamino-p-menthane” 10
75. ”a.a-2.6-tetrachlorotoluene” 13.1
76. ”acenaphthene” 13
77. ”3-methylindole” 15.6
78. ”rotenone #1” 22
79. ”diphenyl phthalate” 8.1
80. ”diethyl phthalate” 8.1
81. ”di-n-butylorthophthalate #1” 8.1
82. ”azinphos-methyl” 22
83. ”salicylanilide” 18
84. ”hexachloro-1.3-butadiene” 2.1
85. ”pentachlorophenol #7” 14.1
86. ”2.4.6-trichlorophenol #1” 14.1
87. ”3-trifluoromethyl-4-nitrophenol” 14.1
88. ”anthranilamide” 8.2
89. ”2-nitrophenol” 14
90. ”2-sec-butyl-4.6-dinitrophenol (dinoseb #1)” 14
91. ”salicylaldehyde” 5
92. ”1-naphthol” 14
93. ”2-phenylphenol” 14
94. ”3.5-dibromosalicylaldehyde” 5
95. ”naphthalene” 13
96. ”quinoline” 15.3
97. ”n.n-diethylcyclohexylamine” 10.2
98. ”n.n-diethylaniline” 10.5
99. ”2-(n-ethyl-m-toluidino)ethanol” 10.5
100. ”1-benzoylacetone” 6.1
101. ”ethyl p-aminobenzoate #2” 8
102. ”piperine (aliphatic)” 15.3
103. ”2.4-dihydroxybenzaldehyde” 5
104. ”o-xylene #1” 13
105. ”o-cresol” 14
106. ”1.2-dichlorobenzene” 13.1
107. ”2-chloroaniline #2” 10.3
108. ”2-fluorotoluene #1” 13.1
109. ”2-chlorophenol #1” 14.1
110. ”1.2.4-trimethylbenzene” 13
111. ”3.4-dichlorotoluene” 13.1
112. ”3.4-dichloroaniline #1” 10.3
113. ”allyl methacrylate” 8.3
114. ”2.3-dibromopropanol” 4
115. ”2-methylbutyraldehyde” 5
116. ”1.2.3-trichloropropane #1” 2
117. ”3-pentanone” 6
118. ”2-butanone oxime” 16
119. ”2-(diisopropylamino)ethanol” 10.2
120. ”2.4-dinitroaniline #1” 10.3
121. ”2.2’-methylenebis(4-chlorophenol)” 14.1
122. ”p-tert-butylphenol” 14
123. ”isopropylbenzene” 13
124. ”acetophenone” 6
125. ”nitrobenzene” 13
126. ”m-aminoacetophenone” 6
127. ”m-nitrotoluene” 13
128. ”n.n-dimethyl-p-toluidine #1” 10.5
129. ”p-nitroaniline” 10.3
130. ”4-nitrophenol #1” 14
131. ”p-dimethylaminobenzaldehyde” 5
132. ”1.4-dinitrobenzene” 13
133. ”n.n-diethylethanolamine” 10.2
134. ”ethylbenzene #1” 13
135. ”benzylamine” 10
136. ”benzaldehyde #1” 5
137. ”n-methylaniline” 10.4
138. ”cyclohexanone oxime” 16
139. ”2-cyanopyridine” 15.3
140. ”2-ethylpyridine” 15.3
141. ”solketal” 3.3
142. ”hexamethylenetetramine (aliphatic)” 15.4
143. ”phenyl ether” 3.1
144. ”n-ethyl-m-toluidine” 10.4
145. ”tripropylamine” 10.2
146. ”triethanolamine” 10.2
147. ”benzyl-tert-butanol” 4
148. ”1-(2-hydroxyethyl)piperazine” 15
149. ”n.n-dimethylbenzylamine” 10.2
150. ”4-acetamidophenol” 18
151. ”4-butylaniline” 10.3
152. ”nonylphenol (mixed)” 14
153. ”2-ethyl-1-hexanol” 4
154. ”4-chlorobenzaldehyde” 5

155. ”5-ethyl-2-methylpyridine” 15.3
156. ”5-diethylamino-2-pentanone” 10.2
157. ”diethyl malonate #1” 8
158. ”2.4-dimethylphenol” 14
159. ”dibutyl fumarate #3” 8
160. ”dibutyl adipate” 8
161. ”p-bromoaniline” 10.3
162. ”p-xylene” 13
163. ”4-methylphenol (p-cresol)” 14
164. ”4-chloroaniline #1” 10.3
165. ”4-chlorophenol” 14.1
166. ”4-toluidine #1” 10.3
167. ”isobutyl acrylate #1” 8.3
168. ”1-bromopropane” 2
169. ”acrolein #1” 5
170. ”1.2-dichloroethane” 2
171. ”2-chloroethanol #5” 4
172. ”propylamine” 10
173. ”propionitrile” 9
174. ”chloroacetonitrile” 9
175. ”ethylenediamine” 10
176. ”allyl alcohol” 4.1
177. ”2-propyn-1-ol #1” 4.2
178. ”acetaldoxime” 16
179. ”2-methyl-2.4-pentanediol” 4.3
180. ”tert-octylamine” 10
181. ”tert-butyl sulfide” 12.1
182. ”2-pentanone” 6
183. ”4-methyl-2-pentanone #2” 6
184. ”isopropyl ether” 3
185. ”toluene #1” 13
186. ”4-picoline” 15.3
187. ”chlorobenzene” 13.1
188. ”cyclohexanol” 4
189. ”cyclohexanone #2” 6.2
190. ”phenol #1” 14
191. ”3-picoline” 15.3
192. ”1-methylpiperazine” 15
193. ”2-picoline” 15.3
194. ”2-methylpiperazine” 15
195. ”propyl acetate” 8
196. ”1.3-dibromopropane #1” 2
197. ”1-bromobutane” 2
198. ”butylamine” 10
199. ”allyl cyanide” 9
200. ”1.3-diaminopropane” 10
201. ”malononitrile (nominals)” 9
202. ”2-methoxyethylamine” 10
203. ”diethylamine” 10.1
204. ”pyrrole” 15.6
205. ”tetrahydrofuran” 3.3
206. ”furan” 3.3
207. ”t-butyl disulfide” 12.2
208. ”5-methyl-2-hexanone” 6
209. ”diethyl sebacate #1” 8
210. ”2-heptanone” 6
211. ”hexane” 1
212. ”1.4-dichlorobutane” 2
213. ”amylamine” 10
214. ”valeraldehyde #1” 5
215. ”2-butyne-1.4-diol” 4.3
216. ”2-(ethylamino)ethanol” 10.1
217. ”cyclohexane” 1
218. ”pyridine #1” 15.3
219. ”s-trioxane” 3.3
220. ”6-methyl-5-hepten-2-one” 6
221. ”2-octanone” 6
222. ”2-ethoxyethyl acetate” 8
223. ”1-bromohexane” 2
224. ”hexylamine” 10
225. ”1-hexanol” 4
226. ”diethanolamine” 10.1
227. ”2-hydroxyethyl ether” 4.3
228. ”n-propyl sulfide” 12.1
229. ”n-heptylamine” 10
230. ”1.4-dicyanobutane” 9
231. ”1-heptanol” 4
232. ”1-bromooctane” 2
233. ”octylamine” 10
234. ”1-octanol #2” 4
235. ”2-(2-ethoxyethoxy)ethanol” 4
236. ”nonanoic acid” 7
237. ”2-undecanone” 6
238. ”nonylamine” 10
239. ”triethylene glycol #1” 4
240. ”1-decanol” 4
241. ”propoxur (baygon)” 21
242. ”2-methyl-3-butyn-2-ol” 4.2
243. ”2.2.2-trichloroethanol” 4
244. ”dicofol (kelthane)” 22
245. ”triphenyl phosphate” 19
246. ”fensulfothion” 22
247. ”aldicarb” 21
248. ”phenyl salicylate” 8
249. ”ethyl salicylate #1” 8
250. ”2.4.6-tribromophenol” 14.1
251. ”4-amino-2-nitrophenol” 14
252. ”benzophenone #2” 6
253. ”n-phenyldiethanolamine” 10.5
254. ”4-(diethylamino)benzaldehyde” 5
255. ”catechol” 14
256. ”1.2.4-trichlorobenzene” 13.1
257. ”2.4-dichlorophenol” 14.1
258. ”2.4-dinitrotoluene” 13
259. ”3-ethoxy-4-hydroxybenzaldehyde” 5

260. ”vanillin #1” 5
261. ”n.n-dimethylaniline #1” 10.5
262. ”1-chloro-3-nitrobenzene” 13.1
263. ”malathion” 22
264. ”2-chloro-4-nitroaniline #2” 10.3
265. ”p-isopropyl benzaldehyde” 5
266. ”diphenylamine” 10.4
267. ”2-phenoxyethanol” 4
268. ”4-ethylphenol” 14
269. ”2-methylvaleraldehyde” 5
270. ”2.4-pentanedione #2” 6.1
271. ”ethyl hexanoate” 8
272. ”butanal #1” 5
273. ”butyl acetate” 8
274. ”1.4-dioxane #1” 3.3
275. ”dodecylamine” 10
276. ”tributyl phosphate #1” 19
277. ”5.5-dimethyl-1.3-cyclohexanedione” 6.1
278. ”1-chloro-2-propanol” 4
279. ”tetrachloroethylene #1” 2.1
280. ”2-phenyl-3-butyn-2-ol” 4.2
281. ”2.6-di-tert-butyl-4-methylphenol” 14
282. ”saccharin sodium salt hydrate” 16
283. ”dibenzofuran #2” 3.3
284. ”phenyl 4-aminosalicylate #1” 8
285. ”n.n-diethyl-m-toluamide” 8.2
286. ”propionic acid. sodium salt” 7
287. ”1-(2-aminoethyl)piperazine” 15
288. ”dibutyl succinate” 8
289. ”diethyl adipate #1” 8
290. ”2-aminoethanol” 10
291. ”ethyl acetate” 8
292. ”m-diethylbenzene” 13
293. ”1.3-dichloropropane #1” 2
294. ”hexanoic acid” 7
295. ”hexyl acetate” 8
296. ”butyl ether” 3
297. ”1-nonanol” 4
298. ”di-n-hexylamine” 10.1
299. ”o-vanillin #1” 5
300. ”3-methoxyphenol” 14
301. ”4-methoxyphenol” 14
302. ”p-dimethoxybenzene” 13
303. ”2.3-benzofuran” 3.3
304. ”1.4-diazabicyclo[2.2.2]octane” 15
305. ”adamantane” 1
306. ”disulfoton #1” 22
307. ”secobarbital. sodium salt” 23
308. ”bromacil” 22
309. ”2.5-dinitrophenol” 14
310. ”diuron” 18
311. ”p-fluorophenyl ether #1” 3.1
312. ”diazinon” 22
313. ”1-fluoro-4-nitrobenzene” 13.1
314. ”a.a.a-trifluoro-m-tolunitrile” 9
315. ”4-fluoroaniline” 10.3
316. ”2-chloro-6-fluorobenzaldehyde” 5
317. ”a.a.a-4-tetrafluoro-o-toluidine” 10.3
318. ”o-fluorobenzaldehyde” 5
319. ”a.a.a-trifluoro-o-tolunitrile” 9
320. ”a.a.a-trifluoro-m-tolualdehyde #1” 5
321. ”4-fluoro-n-methylaniline” 10.4
322. ”[(1s)-endo]-(-)-borneol #2” 4
323. ”(1s)-(-)-camphor” 6.2
324. ”cineole” 3.3
325. ”neoabietic acid #1” 7
326. ”2.3-dihydrobenzofuran” 3.3
327. ”exo-norborneol” 4
328. ”norbornylene” 1.1
329. ”2.6-pyridinedicarboxylic acid” 15.3
330. ”3-pyridinecarboxaldehyde” 15.3
331. ”5-nonanone” 6
332. ”2.3-dimethyl-1.3-butadiene” 1.1
333. ”abietic acid” 7
334. ”flavone” 3.3
335. ”2.4.6-trimethylphenol” 14
336. ”o-tolunitrile” 9
337. ”o-tolualdehyde” 5
338. ”benzoic acid. sodium salt” 7
339. ”4.6-dinitro-o-cresol #1” 14
340. ”amylbenzene” 13
341. ”tert-butyl acetate” 8
342. ”1.3-dichlorobenzene” 13.1
343. ”n-butyl sulfide” 12.1
344. ”2’-hydroxy-4’-methoxyacetophenone #1” 6
345. ”o-nitrobenzaldehyde #2” 5
346. ”4-nitrobenzaldehyde” 5
347. ”3-methyl-2-butanone” 6
348. ”2.6-dinitrophenol” 14
349. ”1.2-dibromobenzene” 13.1
350. ”n-allylaniline” 10.4
351. ”4-ethylaniline” 10.3
352. ”isovaleraldehyde” 5
353. ”2-hexanone” 6
354. ”2-tridecanone” 6
355. ”manool” 4.1
356. ”tetraethyltin” 24
357. ”1.2-dimethylpropylamine” 10
358. ”2.4-dimethyl-3-pentanol” 4
359. ”n.n-diphenylformamide #1” 8.2
360. ”diethyl benzylmalonate” 8
361. ”pentabromophenol” 14.1
362. ”2.4.6-triiodophenol” 14.1
363. ”2.4-dimethoxybenzaldehyde” 5
364. ”2-acetamidophenol #1” 18
365. ”2-chloro-4-methylaniline” 10.3

366. ”4-ethoxy-2-nitroaniline” 10.3
367. ”methyl p-nitrobenzoate” 8
368. ”4-nitrobenzamide” 8.2
369. ”4-nitrophenyl phenyl ether” 3.1
370. ”benzyl sulfoxide” 12.3
371. ”3-acetamidophenol” 18
372. ”4-(2-hydroxyethyl)morpholine” 16
373. ”a.a’-dichloro-p-xylene” 13.1
374. ”2.5-dimethylfuran” 3.3
375. ”1.5-dichloropentane” 2
376. ”1-bromoheptane” 2
377. ”propyl disulfide” 12.2
378. ”1.6-dicyanohexane” 9
379. ”2.3.4-trichloroaniline” 10.3
380. ”5-chlorosalicylaldehyde” 5
381. ”4-propylphenol” 14
382. ”pentafluorobenzaldehyde” 5
383. ”2.2-dichloroacetamide” 8.2
384. ”1-methyl heptylamine #1” 10
385. ”2-decanone #1” 6
386. ”pentyl ether” 3
387. ”4-methyloxazole” 16
388. ”2-methylimidazole” 15.6
389. ”2-adamantanone” 6.2
390. ”a-decanolactone” 8
391. ”4.6-dimethoxy-2-hydroxybenzaldehyde” 5
392. ”propanil” 18
393. ”2.4.6-tri-tert-butylphenol” 14
394. ”3.4-dichloro-1-butene #1” 2.1
395. ”n.n-dibutylformamide” 8.2
396. ”2-butyn-1-ol” 4.2
397. ”2.5-dimethyl-2.4-hexadiene” 1.1
398. ”1-adamantanamine” 10
399. ”3-cyano-4.6-dimethyl-2-hydroxypyridine” 15.3
400. ”2.3.4.5.6-pentafluoroaniline” 10.3
401. ”carbophenothion” 22
402. ”triphenylphosphine oxide” 19
403. ”2-hydroxyethyl acrylate” 8.3
404. ”1-octyn-3-ol” 4.2
405. ”2-nonanone” 6
406. ”trans-1.2-dichlorocyclohexane” 2
407. ”p-phenoxyphenol” 14
408. ”2-hydroxyethyl methacrylate” 8.3
409. ”3-bromothiophene” 17
410. ”2.4-dichlorobenzaldehyde” 5
411. ”phenyl disulfide” 12.2
412. ”ethyl 3-aminobenzoate methanesulfonic acid salt (ms-222)” 8
413. ”1.1.1.3.3.3-hexafluoro-2-propanol” 4
414. ”1.5-hexadien-3-ol” 4.1
415. ”3-butyn-1-ol” 4.2
416. ”cis-3-hexen-1-ol” 4.1
417. ”trans-3-hexen-1-ol” 4.1
418. ”2-acetyl-1-methylpyrrole” 15.6
419. ”4-phenylpyridine” 15.3
420. ”phenyl sulfoxide” 12.3
421. ”2-hydroxypropyl acrylate #1” 8.3
422. ”2-amino-5-bromopyridine” 15.3
423. ”diethyl benzylphosphonate” 19
424. ”4-acetylpyridine” 15.3
425. ”methyl p-chlorobenzoate” 8
426. ”butyl phenyl ether” 3
427. ”methyl 4-cyanobenzoate” 9
428. ”tetrachlorocatechol” 14.1
429. ”a-bromo-2’.5’-dimethoxyacetophenone #1” 6
430. ”tetrabutyltin” 24
431. ”3.4-dimethyl-1-pentyn-3-ol” 4.2
432. ”n-vinylcarbazole” 15.6
433. ”3-benzyloxyaniline” 10.3
434. ”carbofuran” 21
435. ”tert-butyl methyl ether” 3
436. ”1.9-decadiene” 1.1
437. ”p-phenylazophenol” 14
438. ”3.5-diiodo-4-hydroxybenzonitrile” 9
439. ”3.5-dibromo-4-hydroxybenzonitrile #1” 9
440. ”dehydroabietic acid” 7
441. ”2-allylphenol” 14
442. ”t-butylstyrene” 13
443. ”5-bromosalicylaldehyde” 5
444. ”3-chloro-2-chloromethyl-1-propene” 2.1
445. ”3.5-dichloro-4-hydroxybenzonitrile” 9
446. ”di-n-butylterephthalate” 8.1
447. ”4.4’-dihydroxydiphenyl ether #1” 3.1
448. ”2.6-dichlorobenzamide” 8.2
449. ”n-decylamine” 10
450. ”aminocarb” 21
451. ”2.4.5-tribromoimidazole (nominal) #1” 15.6
452. ”o-ethyl o-(p-nitrophenyl phenyl)phosphonothioate” 22
453. ”(+-)-4-pentyn-2-ol” 4.2
454. ”4-chlorocatechol” 14.1
455. ”methyl 2.4-dihydroxybenzoate” 8
456. ”pentachloropyridine” 15.3
457. ”(1r.2s.5r)-(-)-menthol” 4
458. ”1-(p-toluenesulfonyl)imidazole” 15.6
459. ”2’.4’-dichloroacetophenone” 6
460. ”n-octyl cyanide #1” 9
461. ”a.a.a-4-tetrafluoro-m-toluidine” 10.3
462. ”trans-2-phenyl-1-cyclohexanol” 4
463. ”2-ethoxyethyl methacrylate” 8.3
464. ”2.3.6-trimethylphenol” 14
465. ”n-undecyl cyanide” 9

466. ”0-methoxybenzamide” 8.2
467. ”2.4-dichlorobenzamide” 8.2
468. ”tetrahydrofurfuryl methacrylate” 8.3
469. ”4.5-dichloroguaiacol” 14.1
470. ”benzyl methacrylate” 8.3
471. ”hexyl acrylate #1” 8.3
472. ”p-(tert-butyl)-phenyl-n-methylcarbamate” 21
473. ”1-benzylpiperazine” 15
474. ”3-(3-pyridyl)-1-propanol” 15.3
475. ”tridecylamine” 10
476. ”2-amino-4’-chlorobenzophenone” 6
477. ”methyl 2.5-dichlorobenzoate” 8
478. ”chlorpyrifos #2 (dursban)” 22
479. ”5-bromovanillin” 5
480. ”cyclohexyl acrylate” 8.3
481. ”triethyl nitrilotricarboxylate” 21
482. ”4.5-dichlorocatechol” 14.1
483. ”2.3.5.6-tetrachloroaniline” 10.3
484. ”2.6-diphenylpyridine” 15.3
485. ”1.3-dichloro-4.6-dinitrobenzene #1” 13.1
486. ”2.3-dimethylvaleraldehyde” 5
487. ”2-decyn-1-ol” 4.2
488. ”5-chloro-2-pyridinol” 15.3
489. ”isopropyl disulfide” 12.2
490. ”2.4.5-trimethoxybenzaldehyde” 5
491. ”isopropyl methacrylate” 8.3
492. ”1-hexen-3-ol” 4.1
493. ”2.3.4.5-tetrachlorophenol” 14.1
494. ”1.2-bis(4-pyridyl)ethane” 15.3
495. ”1.3-diethyl-2-thiobarbituric acid” 23
496. ”dimethyl nitroterephthalate” 8.1
497. ”5-chloro-2-mercaptobenzothiazole” 16
498. ”dimethyl aminoterephthalate” 8.1
499. ”3.6-dithiaoctane” 12.1
500. ”3-dimethylaminopropyl chloride.hcl” 10.2
501. ”4’-chloro-3’-nitroacetophenone” 6
502. ”2-amino-4-chloro-6-methylpyrimidine #1” 15.2
503. ”2.6-dimethoxytoluene” 13
504. ”2-dimethylaminopyridine” 15.3
505. ”2.2-dimethyl-1-propylamine” 10
506. ”isopimaric acid” 7
507. ”2-amino-5-chlorobenzonitrile” 9
508. ”1.1.1-trichloro-2-methyl-2-propanol(hydrate)” 4
509. ”2-dodecanone” 6
510. ”4-dimethylaminocinnamaldehyde” 5
511. ”1.3.5-trichloro-2.4-dinitrobenzene” 13.1
512. ”2-chloro-5-nitrobenzaldehyde #1” 5
513. ”2-chloro-6-methylbenzonitrile” 9
514. ”2-bromo-3-pyridinol” 15.3
515. ”2-chloro-3-pyridinol” 15.3
516. ”tripropargylamine” 10.2
517. ”n.n-bis(2.2-diethoxyethyl)methylamine #1” 10.2
518. ”1.4-bis(3-aminopropyl)piperazine” 15
519. ”3-hydroxy-3.7.11-trimethyl-1.6.10-dodecatriene” 4.1
520. ”1-(2-chloroethyl)pyrrolidine.hcl” 15.5
521. ”n-undecylamine” 10
522. ”1-heptyn-3-ol” 4.2
523. ”p-ethoxybenzaldehyde” 5
524. ”[1(r)-endo]-(+)-3-bromocamphor” 6.2
525. ”resmethrin” 22
526. ”terbufos (counter)” 22
527. ”a.a.a’.a’-tetrabromo-o-xylene” 13.1
528. ”2’.3’.4’-trichloroacetophenone” 6
529. ”2’.3’.4’-trimethoxyacetophenone #2” 6
530. ”diethyl chloromalonate” 8
531. ”n-ethylbenzylamine” 10.1
532. ”4-bromophenyl 3-pyridyl ketone” 15.3
533. ”4-benzoylpyridine” 15.3
534. ”2.2.5.5-tetramethyltetrahydrofuran” 3.3
535. ”3-hydroxy-2-nitropyridine” 15.3
536. ”alachlor” 18
537. ”4-octylaniline” 10.3
538. ”methomyl (lannate)” 21
539. ”6-chloro-2-pyridinol” 15.3
540. ”3-amino-5.6-dimethyl-1.2.4-triazine” 15.4
541. ”4-(diethylamino)salicylaldehyde” 5
542. ”6-chloro-2-picoline” 15.3
543. ”3.6-dimethyl-1-heptyn-3-ol” 4.2
544. ”2.4.5-trimethyloxazole” 16
545. ”4-dimethylamino-3-methyl-2-butanone” 6
546. ”m-bromobenzamide” 8.2
547. ”oxamyl #1” 21
548. ”2.6-diisopropylaniline #1” 20
549. ”2-methyl-3.3.4.4-tetrafluoro-2-butanol” 4
550. ”chloromethyl styrene” 13.1
551. ”(+-)-sec-butylamine” 10
552. ”n-(3-methoxypropyl)-3.4.5-trimethoxybenzylamine” 10.1
553. ”2-(bromomethyl)tetrahydro-2h-pyran” 3.3
554. ”4-decylaniline” 10.3
555. ”4-hexyloxyaniline #1 (nominal conc.)” 10.3
556. ”methyl 4-chloro-2-nitrobenzoate” 8
557. ”5-hydroxy-2-nitrobenzaldehyde” 5
558. ”fenvalerate #1 (pydrin)” 22
559. ”permethrin” 22
560. ”3.8-dithiadecane” 12.1
561. ”2’-(octyloxy)-acetanilide” 18
562. ”p-(tert-butyl)benzamide” 8.2

563. ”2.9-dithiadecane” 12.1
564. ”dl-3-butyn-2-ol” 4.2
565. ”3-(4-tert-butylphenoxy)benzaldehyde” 5
566. ”flucythrinate” 22
567. ”3-(3.4-dichlorophenoxy)benzaldehyde” 5
568. ”2.4-dinitro-1-naphthol sodium salt (martius yellow)” 14

Code	Class Name	Code	Class Name
1.0	Alkanes	10.5	Tertiary, aromatic amines
1.1	Alkenes	11.1	Azine compounds
2.0	Saturated Hydrocarbons	12.0	Thiols
2.1	Unsaturated Hydrocarbons	12.1	Sulfides
3.0	Basic Ethers	12.2	Disulfides
3.1	Diphenyl Ethers	12.3	Sulfo compounds
3.3	Cyclic Ethers	13.0	Benzenes
4.0	Basic Alcohols	13.1	Chlorinated Benzenes
4.1	Alkene Alcohols	14.0	Phenols
4.2	Alkyne Alcohols	14.1	Chlorinated Phenols
4.3	Diols	15.0	Piperazines
5.0	Aldehydes	15.2	Pyrimidines
6.0	Basic Ketones	15.3	Pyridines
6.1	beta-Diketones	15.4	Triazines
6.2	Cyclic Ketones	15.5	5-Membered ring aliphatics
7.0	Carboxylic Acids	15.6	5-Membered ring aromatics
8.0	Basic Esters	16.0	Multiple hetero-atom compounds
8.1	Phthalates	17.0	Heterocyclic sulfur compounds
8.2	Amides	18.0	Anilides and Ureas
8.3	Acrylates	19.0	Phosphorous compounds
9.0	Nitriles	20.0	Quaternary ammonium compounds
10.0	Primary, aliphatic amines	21.0	Carbamates
10.1	Secondary, aliphatic amines	22.0	Other pesticides
10.2	Tertiary, aliphatic amines	23.0	Barbitals
10.3	Primary, aromatic amines	23.1	DEAS-complex structures
10.4	Secondary, aromatic amines	24.0	Organometallics

Table A.1: Definition of molecule classification codes.

Appendix B

List of Descriptors

1. ”Total Energy (kcal/mol)”, ”QM1”
2. ”Binding Energy (kcal/mol)”, ”QM2”
3. ”Heat of Formation (kcal/mol)”, ”QM3”
4. ”Dipole Moment (D)”, ”QM4”
5. ”HOMO (eV)”, ”QM5”
6. ”LUMO (eV)”, ”QM6”
7. ”Number of atoms”, ”C1”
8. ”Number of C atoms”, ”C2”
9. ”Relative number of C atoms”, ”C3”
10. ”Number of H atoms”, ”C4”
11. ”Relative number of H atoms”, ”C5”
12. ”Number of O atoms”, ”C6”
13. ”Relative number of O atoms”, ”C7”
14. ”Number of N atoms”, ”C8”
15. ”Relative number of N atoms”, ”C9”
16. ”Number of S atoms”, ”C10”
17. ”Relative number of S atoms”, ”C11”
18. ”Number of F atoms”, ”C12”
19. ”Relative number of F atoms”, ”C13”
20. ”Number of Cl atoms”, ”C14”
21. ”Relative number of Cl atoms”, ”C15”
22. ”Number of Br atoms”, ”C16”
23. ”Relative number of Br atoms”, ”C17”
24. ”Number of I atoms”, ”C18”
25. ”Relative number of I atoms”, ”C19”
26. ”Number of P atoms”, ”C20”
27. ”Relative number of P atoms”, ”C21”
28. ”Number of bonds”, ”C22”
29. ”Number of single bonds”, ”C23”
30. ”Relative number of single bonds”, ”C24”
31. ”Number of double bonds”, ”C25”
32. ”Relative number of double bonds”, ”C26”
33. ”Number of triple bonds”, ”C27”
34. ”Relative number of triple bonds”, ”C28”
35. ”Number of aromatic bonds”, ”C29”
36. ”Relative number of aromatic bonds”, ”C30”
37. ”Number of rings”, ”C31”
38. ”Relative number of rings”, ”C32”
39. ”Number of benzene rings”, ”C33”
40. ”Relative number of benzene rings”, ”C34”
41. ”Molecular weight”, ”C35”
42. ”Relative molecular weight”, ”C36”
43. ”Gravitation index (all bonds)”, ”C37”
44. ”Gravitation index (all pairs)”, ”C38”
45. ”Wiener index”, ”T1”
46. ”Randic index (order 0)”, ”T2”
47. ”Randic index (order 1)”, ”T3”
48. ”Randic index (order 2)”, ”T4”
49. ”Randic index (order 3)”, ”T5”
50. ”Kier&Hall index (order 0)”, ”T6”
51. ”Kier&Hall index (order 1)”, ”T7”
52. ”Kier&Hall index (order 2)”, ”T8”
53. ”Kier&Hall index (order 3)”, ”T9”
54. ”Kier shape index (order 1)”, ”T10”
55. ”Kier flexibility index”, ”T13”
56. ”Average Info. content (order 0)”, ”T14”
57. ”Info. content (order 0)”, ”T15”
58. ”Average Structural Info. content (order 0)”, ”T16”
59. ”Structural Info. content (order 0)”, ”T17”
60. ”Average Complementary Info. content (order 0)”, ”T18”
61. ”Complementary Info. content (order 0)”, ”T19”
62. ”Average Bonding Info. content (order 0)”, ”T20”
63. ”Bonding Info. content (order 0)”, ”T21”
64. ”Average Info. content (order 1)”, ”T22”
65. ”Info. content (order 1)”, ”T23”

66. ”Average Structural Info. content (order 1)”, ”T24”
67. ”Structural Info. content (order 1)”, ”T25”
68. ”Average Complementary Info. content (order 1)”, ”T26”
69. ”Complementary Info. content (order 1)”, ”T27”
70. ”Average Bonding Info. content (order 1)”, ”T28”
71. ”Bonding Info. content (order 1)”, ”T29”
72. ”Average Info. content (order 2)”, ”T30”
73. ”Info. content (order 2)”, ”T31”
74. ”Average Structural Info. content (order 2)”, ”T32”
75. ”Structural Info. content (order 2)”, ”T33”
76. ”Average Complementary Info. content (order 2)”, ”T34”
77. ”Complementary Info. content (order 2)”, ”T35”
78. ”Average Bonding Info. content (order 2)”, ”T36”
79. ”Bonding Info. content (order 2)”, ”T37”
80. ”Balaban index”, ”T38”
81. ”Moment of inertia A”, ”G1”
82. ”Moment of inertia B”, ”G2”
83. ”Moment of inertia C”, ”G3”
84. ”XY Shadow”, ”G4”
85. ”XY Shadow / XY Rectangle”, ”G5”
86. ”YZ Shadow”, ”G6”
87. ”YZ Shadow / YZ Rectangle”, ”G7”
88. ”ZX Shadow”, ”G8”
89. ”ZX Shadow / ZX Rectangle”, ”G9”
90. ”Molecular volume”, ”G10”
91. ”Molecular volume / XYZ Box”, ”G11”
92. ”Molecular surface area”, ”G12”
93. ”Max partial charge for a C atom”, ”E3”
94. ”Min partial charge for a C atom ”, ”E4”
95. ”Max partial charge (Qmax) ”, ”E7”
96. ”Min partial charge (Qmin) ”, ”E8”
97. ”Polarity parameter (Qmax-Qmin) ”, ”E9”
98. ”Polarity parameter / square distance ”, ”E10”
99. ”Topographic electronic index (all pairs) ”, ”E11”
100. ”Topographic electronic index (all bonds) ”, ”E12”
101. ”TMSA Total molecular surface area ”, ”E13”
102. ”PPSA-1 Partial positive surface area ”, ”E14”
103. ”PNSA-1 Partial negative surface area ”, ”E15”
104. ”DPSA-1 Difference in CPSAs (PPSA1-PNSA1) ”, ”E16”
105. ”FPSA-1 Fractional PPSA (PPSA-1/TMSA) ”, ”E17”
106. ”FNSA-1 Fractional PNSA (PNSA-1/TMSA) ”, ”E18”
107. ”WPSA-1 Weighted PPSA (PPSA1*TMSA/1000) ”, ”E19”
108. ”WNSA-1 Weighted PNSA (PNSA1*TMSA/1000) ”, ”E20”
109. ”PPSA-2 Total charge weighted PPSA ”, ”E21”
110. ”PNSA-2 Total charge weighted PNSA ”, ”E22”
111. ”DPSA-2 Difference in CPSAs (PPSA2-PNSA2) ”, ”E23”
112. ”FPSA-2 Fractional PPSA (PPSA-2/TMSA) ”, ”E24”
113. ”FNSA-2 Fractional PNSA (PNSA-2/TMSA) ”, ”E25”
114. ”WPSA-2 Weighted PPSA (PPSA2*TMSA/1000) ”, ”E26”
115. ”WNSA-2 Weighted PNSA (PNSA2*TMSA/1000) ”, ”E27”
116. ”PPSA-3 Atomic charge weighted PPSA ”, ”E28”
117. ”PNSA-3 Atomic charge weighted PNSA ”, ”E29”
118. ”DPSA-3 Difference in CPSAs (PPSA3-PNSA3) ”, ”E30”
119. ”FPSA-3 Fractional PPSA (PPSA-3/TMSA) ”, ”E31”
120. ”FNSA-3 Fractional PNSA (PNSA-3/TMSA) ”, ”E32”
121. ”WPSA-3 Weighted PPSA (PPSA3*TMSA/1000) ”, ”E33”
122. ”WNSA-3 Weighted PNSA (PNSA3*TMSA/1000) ”, ”E34”
123. ”RPCG Relative positive charge (QMPOS/QTPLUS) ”, ”E35”
124. ”RPCS Relative positive charged SA (SAMPOS*RPCG) ”, ”E36”
125. ”RNCG Relative negative charge (QMNEG/QTMINUS) ”, ”E37”
126. ”RNCS Relative negative charged SA (SAMNEG*RNCG) ”, ”E38”
127. ”min(#HA; #HD) ”, ”E51”
128. ”count of H-acceptor sites ”, ”E52”
129. ”count of H-donors sites ”, ”E53”
130. ”HA dependent HDSA-1 ”, ”E54”
131. ”HA dependent HDSA-1/TMSA ”, ”E55”
132. ”HA dependent HDSA-2 ”, ”E56”
133. ”HA dependent HDSA-2/TMSA ”, ”E57”
134. ”HA dependent HDSA-2/SQRT(TMSA) ”, ”E58”
135. ”HA dependent HDCA-1 ”, ”E59”
136. ”HA dependent HDCA-1/TMSA ”, ”E60”
137. ”HA dependent HDCA-2 ”, ”E61”
138. ”HA dependent HDCA-2/TMSA ”, ”E62”
139. ”HA dependent HDCA-2/SQRT(TMSA) ”, ”E63”

140. ”HASA-1 ”, ”E64”
141. ”HASA-1/TMSA ”, ”E65”
142. ”HASA-2 ”, ”E66”
143. ”HASA-2/TMSA ”, ”E67”
144. ”HASA-2/SQRT(TMSA) ”, ”E68”
145. ”HACA-1 ”, ”E69”
146. ”HACA-1/TMSA ”, ”E70”
147. ”HACA-2 ”, ”E71”
148. ”HACA-2/TMSA ”, ”E72”
149. ”HACA-2/SQRT(TMSA) ”, ”E73”
150. ”logD pH3”, ”pH3”
151. ”logD pH5”, ”pH5”
152. ”logD pH6.5”, ”pH6.5”
153. ”logD pH7”, ”pH 7”
154. ”logD pH7.4”, ”pH7.4”
155. ”logD pH9”, ”pH9”

Code	Descriptor Class	Number of Descriptors
QM	Quantum-Chemicals	6
C	Constitutional	38
T	Topological	36
G	Geometrical	12
E	Electrostatic	57
PH	Physicochemical	6

Table B.1: Definition of chemical descriptor codes.

Bibliography

[1] Abelson, H.; Sussman, G.J.: *Lisp: a Language for Stratified Design.* Byte Magazine, pp. 207–218, (1988).

[2] Arnold, D.V.; Beyer, H.G.: *Performance analysis of evolution strategies with multi-recombination in high-dimensional $\mathcal{R}^N$-search spaces disturbed by noise.* Theoretical Computer Science, vol.289(1), pp.629–647, (2002).

[3] Babuska, R.: *Fuzzy Modeling and Identification.* PhD thesis, Dept. of Control Engineering, Delft University of Technology, Delft, The Netherlands, (1996).

[4] Babuska, R.: *Fuzzy Modeling for Control.* Kluwer Academic Publishers, Boston, (1998).

[5] Bäck, T.; Hoffmeister, F.: *Extended Selection Mechanisms in Genetic Algorithms.* Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo, California, USA: Morgan Kaufmann Publishers, pp.92–99, (1991).

[6] Bäck, T.: *Selective pressure in evolutionary algorithms: A characterization of selection mechanisms.* Proceedings of First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence (ICEC94), pp.57-62, (1994).

[7] Baker, J. E.: *Reducing Bias and Inefficiency in the Selection Algorithm.* Proceedings of the Second International Conference on Genetic Algorithms and their Application, Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates,pp. 14–21, (1987).

[8] Backus, J.W.; Bauer, F.L.; Green, J.; Katz, C.; McCarthy, J.; Naur, P.; Perlis, A.J.; Rutishauser, H.; Samuelson, K.; Vauquois, B.; Wegstein, J.H.; van Wijngaarden, A.; Woodger, M.: Revised Report on the Algorithmic Language Algol 60, originally edited by Peter Naur, (1963).

[9] Bardossy, A.; Duckstein, L.; *Fuzzy Rule-Based Modeling with Application to Geophysical, Biological and Engineering Systems.* CRC Press, Boca Raton, Florida, (1994).

[10] Bean, J.C.; Hadj-Alouane, A.B.: *A Dual Genetic Algorithm for Bounded Integer Programs.* Department of Industrial and Operations Engineering, University of Michigan, Technical Report 92-53, (1992).

[11] Benfenati, E.: `mailto:benfenati@marionegri.it`

[12] Blickle, T.; Thiele, L.: *A Comparison of Selection Schemes used in Genetic Algorithms.* Technical Report 11, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zürich, Gloriastrasse 35, CH-8092 Zürich, (1995).

[13] Blickle, T.; Thiele, L.: *A Mathematical Analysis of Tournament Selection.* In Eshelman, L. (editor): Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA95), Morgan Kaufmann, San Francisco, CA,, (1995).

[14] Borgelt, C.; Kruse, R.: *Graphical Models - Methods for Data Analysis and Mining.* ISBN: 0470843373, Wiley&Sons, (2002).

[15] Breiman, L.; Stone, J.H.; Friedman, J.H.; Olshen, R.: *Classification and Regression Trees.* Chapman & Hall, New York, (1984).

[16] Breiman, L.; Spector, P.: *Submodel selection and evaluation in regression: The X-random case.* International Statistical Review, vol.60, pp.291–319, (1992).

[17] Breiman, L.: *Heuristics of instability and stabilization in model selection.* Annals of Statistics, vol.24, pp.2350-2383, (1996).

[18] Brown, M.; Harris, C.J.: *Neurofuzzy Adaptive Modelling and Control.* Prentice Hall, New York, (1994).

[19] Bulmer, M.G.: *The Mathematical Theory of Quantitative Genetics.* Clarendon Press, Oxford, (1980).

[20] Burman, J.D.; Raftery, A.E.: *Model-based Gaussian and non-Gaussian clustering.* Biometrics, vol.49 pp.803–821, (1993).

[21] Cellier, F.E.: *General Systems Problem Solving Paradigm for Qualitative Modeling.* In: Fishwick, P.A., Luker, P.A., (eds.), Qualitative Simulation Modeling and Anaysis, Springer, New York, pp.51–71, (1991).

[22] Chen, S.; Billings, S.A.; Luo, W.: *Orthogonal least squares methods and their application to nonlinear system identification.* International Journal of Control, vol.50(5), pp.1873–1896, (1989).

[23] Chen, S.; Cowan, C.F.N.; Grant, P.M.: *Orthogonal least-squares learning algorithm for radial basis function networks.* IEEE Transactions on Neural Networks, vol.2(2), (1991).

[24] Chen, D.; Giles, C.L.; Sun, G.Z.; Chen, H.H.; Lee, Y.C.; Goudreau, M.W.: *Constructive learning of recurrent neural networks.* Proceedinds IEEE International Conference on Neural Networks, vol.3, pp.1196–1201, San Francisco, (1993).

[25] Cordón, O.; Herrera, F.; Hoffmann, F.; and Magdalena, L.: *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases.* Advances in Fuzzy Systems - Application and Theory vol.19, World Scientific Publishing Co. Pte. Ltd., ISBN 981-02-4016-3, p.337, (2001).

[26] Cordón, O.; Herrera, F.; Hoffmann, F.; and Magdalena, L.: *Genetic Fuzzy Rule-Based Systems Based on the Michigan Approach.* Chapter 6, pp.153–178 in [25], (2001).

[27] Cordón, O.; Herrera, F.; Hoffmann, F.; and Magdalena, L.: *Genetic Fuzzy Rule-Based Systems Based on the Pittsburg Approach.* Chapter 7, pp.179–218 in [25], (2001).

[28] Couvreur, C.; Bresler, Y.: *On The Optimallity Of The Backward Greedy Algorithm For The Subset Selection Problem.* SIAM Journal on Matrix Analysis and Applications, (1993).

[29] Cox, M.G.: *The numerical evaluation of B-splines.* Journal of Mathematics and its Applications, vol.10, pp.134–149, (1972).

[30] Darwin, C.R.: *The Origin of Species.* The Harvard Classics, Ed. Eliot, C.W., vol. XI, P.F. Collier & Son, 1909–14, New York, (1859).

[31] Deb, K.: *Multi-Objective Optimization using Evolutionary Algorithms.* Publisher: John Wiley&Sons, ISBN: 047187339X, (2001).

[32] de Boor, C.: *On calculating with B-splines.* Journal of Approximation Theory, vol.6, pp.50–62, (1972).

[33] de Castro, L.N.; Timmis, J.: *Artificial Immune Systems: A New Computational Intelligence Approach.* Publisher Springer, (2002).

[34] De Jong, K.A.: *Genetic Algorithms: A 10 Year Perspective.* Editor: Grefenstette, J.J., Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, pp.169–177, (1985).

[35] De Jong, K.A.: *On Using Genetic Algorithms to Search Program Spaces.* Editor: Grefenstette, J.J., Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, pp.169–177, (1987).

[36] De Jong, K.A.: *Learning with Genetic Algorithms: An Overview.* Machine Learning, vol.3, pp.121–138, (1988).

[37] Driankov, D.; Hellendoorn, H.; Reinfrank, M.: *An introduction to Fuzzy Control.* ISBN:0-387-56362-8, Springer Verlag, (1993).

[38] Dierckx, P.: *Curve and Surface Fitting with Splines.* Monographs on numerical analysis, Oxford University press, (1993).

[39] De Jong, K.A.; Spears, W.M.: *Using Genetic Algorithms to Solve NP-Complete Problems.* Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp.124-132, (1989).

[40] Doolittle, W.F.: *Genes in pieces: Where they ever together?* Nature, 272:581, (1978).

[41] Du, D.; Gu, J.; P.P.; (Editors): *Satisfiability Problem: Theory and Applications.* AMS, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol.35, (1997).

[42] Dubois, D.; Prade, H.: *Fuzzy Sets and Systems: Theory and Applications.* New York, Academic Press, (1980).

[43] Dubois, D.; Prade, H.: *A review of fuzzy set aggregation connectives.* Information Science vol.36 pp.85–121, (1985).

[44] ECOTOXicology Database System: *Code List.* Prepared for U.S. Enviromental Protection Agency, Office of Research, Laboratory Mid-Continent Division (MED), Duluth, Minnesota. By OAO Corporation Duluth, Minnesota, (2000).

[45] ECOTOXicology Database System: *Data Field Definition.* Prepared for U.S. Enviromental Protection Agency, Office of Research, Laboratory Mid-Continent Division (MED), Duluth, Minnesota. By OAO Corporation Duluth, Minnesota, (2000).

[46] ECOTOXicology Database System: *User Guide.* Prepared for U.S. Enviromental Protection Agency, Office of Research, Laboratory Mid-Continent Division (MED), Duluth, Minnesota. By OAO Corporation Duluth, Minnesota, (2000).

[47] Efron, B.: *Bootstrap methods: Another look at the jackknife.* Ann. Statist., vol.7, pp. 1–26, (1979).

[48] Efron, B.: *The Jackknife, the Bootstrap and Other Resampling Plans.* Philadelphia: SIAM, (1982).

[49] Eiben, A.E.; van Kemenade, C.H.M.: *Performance of Multi-Parent Crossover Operators on Numerical Function Optimization Problems.* Technical Report 95-33, Rijksuniversiteit te Leiden, Vakgroep Informatica, Leiden, (1995).

[50] Eriksson, L.; Johansson, E.; Müller, M.; Wold, S.: *On the selection of the training set in environmental QSAR analysis when compounds are clustered.* Journal of Chemometrics Volume 14, Special issue: Proceedings of the SSC6, Norway edited by Prof. K. Esbensen,pp. 599–616, (2000).

[51] Fahlman, S.E.; Lebiere, C.: *The cascade-correlation learning architecture.* In Touretzky, D.S., ed., Advances in Neural Information Processing Systems II, San Mateo, CA: Morgan Kaufmann, pp.524–532, (1990).

[52] Filho, J.L.R.; Alippi, C.; Treleaven, P.C.: *Parallel Genetic Algorithms: Theory and Applications - Genetic Algorithm Programming Environments.* IOS Press, Ed. Stender, J., Amsterdam, pp.65–83, (1993).

[53] Fogel, L.J.; Owens, A.J.; Walsh, M.J.: *Artificial Intelligence through Simulated Evolution.* New York: John Wiley & Sons, (1966).

[54] Frean, M.: *The upstart algorithm: A method for constructing and training feed-forward neural networks.* Neural Computation, vol.2(2), pp.198–209, (1990).

[55] Friedman, J.H.; Stuetzle, W.: *Projection pursuit regression.* Journal of the American Statitic Association, vol.76, pp.817–823, (1981).

[56] Friedman, J.H.: *Multivariate adaptive regression splines (with discussion).* The Annals of Statistics, vol.19(1), pp.1–141, (1991).

[57] Garey, M.R.; Johnson, D.S.: *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman & Company, San Francisco, (1978).

[58] Gilbert, W.: *Why genes in pieces?* Nature, 271:501, (1978).

[59] Gilbert, W.: *The exon theory of genes.* Cold Spring Harbor Sym. on Quantitative Biology, vol.52, pp.901–905, (1987).

[60] GLib (Software): GLib Reference Manual.

[61] Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley, (1989).

[62] Goldberg, D.E.; Deb, L.; Korb, D.: *An investigation of Messy Genetic Algorithms.* Technical Report TCGA 90005, TCGA, University of Alabama, USA, (1990).

[63] Goldberg, D. E.; Deb, K.: *A Comparative Analysis of Selection Schemes Used in Genetic Algorithms.* Foundations of Genetic Algorithms, San Mateo, California, USA: Morgan Kaufmann Publishers, pp.69–93, (1991).

[64] Gottlieb, J.; Raidl, G.R.: *The Effects of Locality on the Dynamics of Decoder-Based Evolutionary Search.* Proceedings of the Genetic and Evolutionary Computation Conference, pp.283–290, (2000).

[65] Goutte, C.: *Note on free lunches and cross-validation.* Neural Computation, vol.9, pp.1211–1215, (1997).

[66] Grauel, A.; Mackenberg, H.: *Mathematical analysis of the Sugeno controller leading to general design rules.* Fuzzy Sets and Systems, vol.85, pp.165–175, (1997).

[67] Grauel, A.; Ludwig, L.A.; Klene, G.: *Comparison of different intelligent methods for process and quality monitoring.* International Journal of Approximate Reasoning, vol.16, Issue 1, pp.89–117, (1997).

[68] GTK+ (Software): GTK+ Reference Manual.

[69] Hall, L.O.; Ozyurt, B.; Bezdek, J.C.: *Clustering with a Genetically Optimized Approach.* IEEE Transactions on Evolutionary Computation, vol.3, number 2, pp.103–112, (1999).

[70] Harvey, I.: *The Artificial Evolution of Adaptive Behaviour.* D. Phil. Thesis, School of Cognitive and Computing Science, University of Sussex, Brigthon, England, (1993).

[71] Hertz, A.; Taillard, E.; de Werra, D.: *A Tutorial On Tabu Search.* Proceedings of Giornate di Lavoro AIRO'95 (Enterprise Systems: Management of Technological and Organizational Changes), (1992).

[72] Hinterding, R.: *Representation, Constraint Satisfaction and the Knapsack Problem.* Proceedings of the 1999 IEEE Congress on Evolutionary Computation, pp. 1286–1292, (1999).

[73] Holland, J.H.: *Adaption in Natural and Artificial Systems.* The University of Michigan Press, USA, (1975).

[74] Homaifar, A.; Lai, S.H.Y.; Qi, X.: *Constrained Optimization via Genetic Algorithms.* Simulation, vol.62, pp.242–254, (1994).

[75] Höppner, F.; Haas. R.; Kruse, R.: *Fuzzy Clusteranalyse.* Vieweg, Braunschweig, (1997).

[76] Hp̈pner, F.; Klawonn, F.: *Improved Fuzzy Partitions for Fuzzy Regression Models.* International Journal of Approximate Reasoning, vol.32, pp.85–102, (2003).

[77] Hunt, K.J.; Haas, R.; Murray-Smith, R.: *Extending the functional equivalence of radial basis function networks and fuzzy inference systems.* IEEE Transactions on Neural Networks, vol.7(3), pp.776–781, (1996).

[78] Hurvich, C.M.; Tsai, C.-L.: *Regression and time series model selection in small samples.* Biometrika, vol.76, pp.297–307, (1989).

[79] *http://<IMAGETOX - Intelligent Modelling Algorithms for General Evaluation of TOXicities.>* Contract Number: HPRN-CT-1999-00015, (1/2/2000–31/1/2004).

[80] Huwendiek, O.; Brockmann, W.: *Der NetFAN-Ansatz als universeller Funktionsapproximator.* In: Grauel, A.; Becker, W.; Belli, F.: (Hrsg.), Fuzzy-Neuro-Systeme'97, pp.324–331, Sankt Augustin, Germany, (1997).

[81] Huwendiek, O.; Brockmann, W.: *Modellierung nichtlinearer Systeme mit dem NetFAN-Ansatz.* 9. GMA-Workshop "Fuzzy Control", Forschungsbericht Nr. 0499, 92-105, Universität Dortmund, Dortmund, Germany, (1999).

[82] Hüllermeier, E.; Renners, I.; Grauel, A.: *An Evolutionary Approach to Constraint-Regularized Learning.* Submitted to a special issue of the journal Mathware and Soft Computing, to appear in (2004).

[83] *http://<The Latest CAS Registry Number® and Substance Count.>*

[84] *http://<MedChem/Biobyte QSAR database.>*

[85] *http://<Multicriterion problem and Pareto optimality.>*

[86] *http://<List of Research Groups in Logic and Theoretical Computer Science World-wide.>*

[87] *http://<Example of a delpoyment flowchart.>*

[88] *http://<Example of an opportunity flowchart.>*

[89] *http://<Molecular Biologists: James Watson & Francis Crick.>*

[90] *http://<Small bibliography Jean-Baptiste Lamarck (1744-1829).>*

[91] Isermann, R.: *Identifikation dynamischer Systeme 1.* ISBN 3-540-54924-2, Springer Verlag, (1991).

[92] Ishibushi, H.; Nakashima, T.; Murata, T.: *Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems.* IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics, vol.29(5), pp.601–618, (1999).

[93] Jang, S.R.; Sun, C-T.: *Functional equivalence between radial basis function networks and fuzzy inference systems.* IEEE Transactions on Neural Networks, vol.4(1), pp.156–159, (1993).

[94] Joines, J.; Houck, C.: *On the Use of Non-stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GAs.* Proceedings of the 1994 IEEE Conference on Evolutionary Computation, IEEE Press, Piscataway, NJ, pp.579–584, (1994).

[95] Jupp, D.L.B.: *Approximation to data by splines with free knots.* SIAM Journal on Numerical Analysis, vol.15, pp.328–343, (1978).

[96] Katritzky, A.R.; Petrukhin, R.; Yang, H.; Karelson, M.: *CODESSA PRO Comprehensive Descriptors for Structural and Statistical Analysis.*,(2001).

[97] Kavli, T.: ASMOD - an Algorithm for Adaptive Spline Modeling of Observation Data. Advances in Intelligent Control, Ed Harris C.J., Taylor and Francis, London, Chapter 6, (1994).

[98] Kecman, V.; Pfeiffer, B-M.: *Exploiting the structural equivalence of learning fuzzy systems and radial basis function networks.* European Congress on Intelligent Techniques and Soft Computing (EUFIT), pp.58–66, Aachen, Germany, (1994).

[99] Klir, G.J.; Folger, T.A.: *Fuzzy Sets, Uncertainty and Information.* Englewood Cliffs, NJ: Prentice-Hall, (1988).

[100] Klir, G.J.: *Facets of Systems Science.* ISBN: 030643959X, Plenum Pub Corp, (1991).

[101] Kohavi, R.: *A study of cross-validation and bootstrap for accuracy estimation and model selection.* International Joint Conference on Artificial Intelligence (IJCAI), (1995).

[102] Kolmogorov, A.N.: *Logical Basis for Information Theory and Probability Theory.* IEEE Transactions on Information Theory, vol.IT-14, no.5, pp.662–664, (1968).

[103] Koza, J.R.: *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems.* Report No. STAN-CS-90-1314, Stanford University, (1990).

[104] Koza, J.R.: *The Genetic Programming Paradigm: Genetically Breeding Populations of Computer Programs to Solve Problems.* Dynamic, Genetic, and Chaotic Programming. The Sixth Generation, New York: John Wiley & Sons, pp.203–321, (1992).

[105] Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, Massachusetts, USA, (1992).

[106] Koziel, S.; Michalewicz, Z.: *Parallel Problem Solving from Nature – PPSN V: A Decoder-Based Evolutionary Algorithm for Constrained Parameter Optimization Problems.* Lecture Notes in Computer Science, Springer, Berlin, pp.231–240, (1998).

[107] Kruse, R.; Gebhardt, J.; Klawonn, F.: *Fuzzy Systeme.* Leitfäden der Informatik, Teubner, 2. überarbeitete Auflage, Kapitel 2.6, pp. 43–57, (1995).

[108] Lee, C.C.: *Fuzzy logic in control systems: Fuzzy logic controllers, Part I and II.* IEEE Transactions on Systems, Man, and Cybernetics, vol.20(2), pp.404–435, (1990).

[109] Li, M.; Vitanyi, P.: *An Introduction to Kolmogorov Complexity and Its Applications.* ISBN 0-387-94868-6, Second Edition, Springer Verlag, (1997).

[110] Lin, C.T.; Lee, C.S.G.: *Neural Fuzzy Systems - A Neuro-Fuzzy Synergism to Intelligent Systems.* Prentice Hall, (1996).

[111] Lin, C.T.; Lee, C.S.G.: In [110], p.25, (1996).

[112] Lin, C.T.; Lee, C.S.G.: In [110], chapter 6.3: Approximate Reasoning, pp.123–130.

[113] Lin, C.T.; Lee, C.S.G.: In [110], chapter 17.2.2: Equivalence of Simplified Fuzzy Inference Systems and Radial Basis Function Networks, pp.487–489.

[114] Lucic, B.; Basic, I.; Nadramija, D.; Milicevic, A.; Trinajstic, N.; Suzuki, T.; Petrukhin, R.; Karelson, M; Katritzky, R.: *Correlation of liquid viscosity with molecular structure for organic compounds using different variable selection methods.* ISSN 1424-6376, Issue in honor of professor Dionis Sunko, ARKIVOC (IV) pp.45–59, (2002).

[115] Ludwig, L.A.: *Computational Intelligence in der Produktionswirtschaft.* HNI-Verlagsschriftenreihe, ISBN: 3931466728, (2000).

[116] Mamdani, E.H.; Gaines, B.R. (eds.): *Fuzzy Reasoning and its Applications.* London, Academic Press, (1981).

[117] Marchand, M.; Golea, M.; Ruján, P.: *A convergence theorem for sequential learning in two-layer perceptrons.* Europhys. Lett. vol.11, pp.487–492, (1990).

[118] *Environmental Chemistry and Toxicology Laboratory at Istituto Mario Negri, via Eritrea 62, 20157 Milan, Italy.*

[119] Martens, H.A.; Lyngby; Dardenne, P.: *Validation and verification of regression in small data sets.* Chemometrics and Intelligent Laboratory Systems Vol. 44, Nos. 1,2, Special Issue: A Collection of Papers Presented at the Fifth Scandinavian Symposium on Chemometrics, Lahti, Finland, edited by Pentti Minkkinen, pp. 99–122, (1998).

[120] Masters, T.: *Advanced Algorithms for Neural Networks: A C++ Sourcebook.* John Wiley and Sons, ISBN 0-471-10588-0, NY, (1995).

[121] Mayer, H.A.: *ptGAs - Genetic Algorithms Evolving Non-Coding Segments by means of Promotor/Terminator Sequences.* Evolutionary Computation vol.6(4), MIT Press, pp.361–386, (1999).

[122] Mazzatorta, P.; Benfenati, E.; Neagu, D.; Gini, G.: *The Importance of Scaling in Data Mining for Toxicity Prediction.* Journal of Chemical Information and Computer Sciences 42(5): pp.1250–1255, (2002)

[123] Mazzatorta, P.; Benfenati, E.; Neagu, D.; Gini, G.: *Tuning Neural and Fuzzy-Neural Networks for Toxicity Modeling.* Journal of Chemical Information and Computer Sciences 43(2), pp.513–518, (2003).

[124] Mazzatorta, P.; Vracko, M.; Jezierska, A.; Benfenati, E.: *Modeling Toxicity by Using Supervised Kohonen Neural Networks.* Journal of Chemical Information and Computer Sciences 43(2), pp.485–492, (2003).

[125] Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs.* Third, Revised and Extented Edition, ISBN 3-540-60676-9, Springer, pp.209–237, (1996).

[126] Michalewicz, Z.: *Decoders* In *Handbook of evolutionary computation*, Oxford University Press, chapter 5.3, (1997).

[127] Michalewicz, Z.; Fogel, D.B.: *How to Solve It: Modern Heuristics.* Corrected Second Printing, ISBN 3-540-666061-5, Pub. Springer, pp.36–37, (2000).

[128] Mitaim S.; Kosko, B.: *What is the Best Shape for a Fuzzy Set in Function Approximation?* In IEEE International Conference on Fuzzy Systems, pp.1237–1243, (1996).

[129] Mooney, C. Z.; Duval, R. D.: *BOOTSTRAPPING A Nonparametric Approach to Statistical Inference.* Series: Quantitative Applications in the Social Sciences. Newbury Park: Sage, (1993).

[130] Nadal, J.: *Study of a growth algorithm for neural networks.* International Journal on Neural Systems, vol.1, pp.55–59, (1989).

[131] Nelles, O.: *Automatische Strukturselektion für Fuzzy-Modelle zur Identifikation nichtlinearer, dynamischer Prozesse.* Oldenbourg Verlage, Automatisierungstechnik, vol.6, (1998).

[132] Nelles, O.: *Nonlinear System Identification - From Classical Approaches to Neural Networks and Fuzzy Models.* ISBN 3-540-67369-5, Springer, (2001).

[133] Nelles, O.: *Loss Functions for Supervised Methods.* In [132], chapter 2.3, pp.28–30, (2001).

[134] Nelles, O.: *Regularization: Implicit Structure Optimization.* In [132], chapter 7.5, pp.179–189, (2001).

[135] Nelles, O.: *Look-Up Table Models.* In [132], chapter 10.3, p.224, (2001).

[136] Nelles, O.: *Local estimation.* In [132], chapter 13.2.2, pp.352–356, (2001).

[137] Nelles, O.: *Dynamic Local Linear Neuro-Fuzzy Models.* In [132], chapter 20, pp.601–644, (2001).

[138] Oliver, I.M.; Smith, D.J.; Holland, J.R.C.: *A study of permutation crossover operators on the TSP.* In Genetic Algorithms and Their Applications: Proceedings of the Second International Conference, Cambridge, MA, pp.224–230, (1987).

[139] O'Reilly, U.-M..; Oppacher, F.: *The troubling Aspects of a building block hyphthesis for genetic programming.* In Whitley, L.D.; Vose M.D.: (editors), Foundations of Genetic Algorithms 3. Morgan Kaufman, (1995).

[140] Palmer, C.C.; Kershenbaum, A.: *Representing Trees in Genetic Algorithms.* Proceedings of the First IEEE International Conference on Evolutionary Computation, Editors: Michalewics, Z.; Schaffer, D.; Schwefel, H.-P.; Fogel, D.; Kitano, H., IEEE Service Center, Piscataway, NJ, USA, vol.1, pp.379-384, (1994).

[141] Pardalos, P.: *On the Passage from Local to Global in Optimization.* In Mathematical Programming, Birge, J.R. and Murty, K.G., eds, University of Michigan, (1994).

[142] Peña-Reyes, C.A.; Sipper, M.: *A fuzzy-genetic approach to breast cancer diagnosis.* Artificial Intelligence in Medicine, vol.17(2), pp.131–155, (1999).

[143] Poli, R.; McPhee, N.F.: *Exact GP Schema Theory for Headless Chicken Crossover and Subtree Mutation.* Proceedings of the Congress on Evolutionary Computation (CEC), (2001).

[144] Poon, P.W.; Carter, J.N.: *Genetic algorithm crossover operators for ordering applications.* Computers & Operations Research vol.22(1), pp.135–148, (1995).

[145] Pospelov, G,S.: *Fuzzy set theory in the USSR.* Fuzzy Sets and Systems, vol.22, pp.1–24, (1987).

[146] Pedrycz, W. (Editor): *Fuzzy Evolutionary Computation.* ISBN 0-7923-9942-0, Kluwer Academic Publishers, Boston, London, Dordrecht, (1997).

[147] Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P: *Numerical Recipes in C: The Art of Scientific Computing.* Second Edition, Cambridge University Press, (1996).

[148] Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P: *Cholesky Decomposition.* Chapter 2.9 in [147], (1996).

[149] Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P: *QR Decomposition.* Chapter 2.10 in [147], (1996).

[150] Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P: *Simulated Annealing Methods.* Chapter 10.9 in [147], (1996).

[151] Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P: *Reduction of a Symmetric Matrix to Tridiagonal Form: Givens and Householder Reductions.* Chapter 11.2 in [147], (1996).

[152] Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P: *Linear Regularization Methods.* Chapter 18.5 in [147], (1996).

[153] Quinlan, J.: *C4.5 Programs for Machine Learning.* Morgan Kaufmann Publishers, San Francisco, (1993).

[154] Rechenberg, I.: *Evolutionsstrategie. Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Stuttgart: Frommann-Holzboog, (1973).

[155] Reed, R.: *Pruning Algorithms: A survey.* IEEE Transactions on Neural Networks, vol.4(4), pp.740–747, (1993).

[156] Rumelhart, D.E.; Hinton, G.E.; Williams, R.J.: *Learning internal representations by error propagation.* Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol.1, chapter 8, New York, USA, (1986).

[157] Renners, I. and Grauel, A. *Optimizing lattice-based associative memory networks by evolutionary algorithms.* Information Sciences, Recent Advances in Genetic Fuzzy Systems, vol. 136, pp. 69–84, (2001).

[158] Renners, I.; Grauel, A.; Saavedra, E.: *Methodology for Optimizing Fuzzy Classifiers Based on Computational Intelligence.* Lecture Notes in Computer Science, Springer-Verlag Heidelberg, ISSN: 0302-9743, Volume 2206 / 2001, Computational Intelligence. Theory and Applications : Proc. of Int. Conf. 7th Fuzzy Days Dortmund, Reusch, B. (Ed.), Germany, (2001).

[159] Renners, I.; Grauel, A.; Ludwig, L.A.; Benfenati, E.; Pelegatti. S.; Robert. D.; Carbó-Dorca, R.; Girones, X.: Modelling Toxicity with Molecular Descriptors and Similarity Measures via B-Spline Networks. Proc. IPMU2000, 8th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge Based Systems, Madrid, Spain, pp.1021–1026, (2000).

[160] Renners, I. and Grauel, A. *Variable length coding of genotypes using n-ary trees.* Proceedings of the 9th International Conference on Soft Computing – Mendel, (2003).

[161] Ruelle, D.: *Zufall und Chaos.* ISBN: 3540577866, 2. Auflage, Springer Verlag, (1994).

[162] Russom, C.L.; Bradbury, S.P.; Broderius, S.J.; Hammermeister, D.E.; Drummond, S.J.: *Predicting modes of toxic action from chemical structure: Acute Toxicity in the fathead minnow (pimephales promelas).* Environ. Toxcicol. Chem. vol.16, pp.948–967, (1997).

[163] Sanger, T.D.: *A tree-structured adaptive network for function approximation in high-dimensional spaces.* IEEE Transactions on Neural Networks, vol.2(2), pp.285–293, (1991).

[164] Schwefel, H.P.: *Evolutionsstrategie und numerische Optimierung.* Dissertation, TU Berlin, (1975).

[165] Shao, J.: *Linear model selection by cross-validation.* Journal of the American Statistical Association, vol.88, pp.486–494, (1993).

[166] Shao, J.; Tu, D.: *The Jackknife and Bootstrap.* New York: Springer-Verlag, (1995).

[167] Shapiro, J.; Prügel-Bennet, A: *A Statistical Mechanical Formulation of the Dynamics of Genetic Algorithms.* In ”Evolutionary Computing”, LNCS 865, Publisher: Springer, Berlin, pp.17–27, (1994).

[168] Shimojima, K.; Kubota, N.; Fukuda, T: *RBF fuzzy controller with virus-evolutionary genetic algorithm.* In Proceedings of the 1996 IEEE International Conference on Neural Networks, vol.2, pp.1040–1043, Washington, DC, IEEE, New York, USA, (1996).

[169] Shipman, R.; Shackleton, M.; Ebner, M.; Watson, R.: *Neutral Search Spaces for Artificial Evolution: A Lesson from Life.* In Bedau, M.A.; Rasmussen, S.; McCaskill, J.S.; Packard, N.H.: (editors), Artificial Life: Proceedings of the Seventh International Conference on Artificial Life, MIT Press,pp.162—169, (2000).

[170] Smith, A.; Tate, D.: *Genetic Optimization Using A Penalty Function.* Proceedings of the Fifth International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp.499–503, (1993).

[171] Smyth, P.: *Clustering using Monte Carlo Cross-Validation.* Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, OR, AAAI Press, pp.126–133, (1996).

[172] Stone, M.: *Cross-validatory choice and assessment of statistical predictions.* Journal of the Royal Statistical Society vol.36, pp.111–147, (1974).

[173] Stone, M.: *Asymptotics for and against cross-validation.* Biometrika, vol.64, pp.29–35, (1977).

[174] Stone, M.: *An Asymptotic Equivalence of Choice of Model-Validation and Akaikes Criterion.* Journal of the Royal Statistical Society, Series B, 39, pp.44–47, (1977).

[175] Stone, M.: *Comments on model selection criteria of Akaike and Schwarz.* Journal of the Royal Statistical Society, series B, vol.41, pp.276–278, (1979).

[176] Sugeno, M.: *An introductory survey of fuzzy control.* Information Science, vol.36, pp.59–83, (1985).

[177] Syswerda, G.: *Uniform Crossover in Genetic Algorithms.* Proceedings of the Third International Conference on Genetic Algorithms, San Mateo/CA: Morgan Kaufmann, pp.2–9, (1989).

[178] Tacket, W.A.: *Recombination, Selection, and the Genetic Construction of Computer Programs.* Ph.D. thesis, Department of Computer Engineering, University of Southern California, (1994).

[179] Tanaka, K.; Sano, M.; Watanabe, H.: *Modeling and Control of Carbon Monoxide Concentration Using a Neuro-Fuzzy Technique.* IEEE Trans. Fuzzy Systems, vol.3, pp.271–279, (1995).

[180] Tanaka, K.; Sugeno, M.: *Introduction to Fuzzy Modeling.* Fuzzy Systems Modeling and Control (Nguyen, H.T. and Sugeno, M., Eds.), Kluwer Academic Publ., Boston, MA, pp.63-89, (1998).

[181] Tibshirani, R.: *A comparison of some error estimates for neural network models.* Neural Computation, vol.8, pp.152–163, (1996).

[182] Tsutsui, S.; Jain, L.C.: *On the Effect of Multi-parents recombination in Binary Coded Genetic Algorithms.* Second International Conference on Knowledge-Based Intelligent Electronic Systems (KES-98), (1998).

[183] Turney, P.: *Myths and legends of the Baldwin Effect.* Proceedings of the 13th International Conference on Machine Learning, (1996).

[184] Unbehauen, R.: *Systemtheorie.* ISBN: 3-486-21513-2, 5. Auflage, Oldenburg Verlag, (1990).

[185] Von Bertalanffy, L.: *General System Theory: Foundations, Development, Applications.* ISBN: 0807604534, Braziller, G. (Publisher), Revised edition, (1976).

[186] Werbos, P.J.: *New Tools for Prediction and Analysis in the Behavioural Sciences.* PhD thesis, Harvard University, Boston, USA, (1974).

[187] Werntges, H.W.: *Partitions of unity improve neural function approximators.* IEEE International Conference on Neural Networks (ICNN), San Francisco, USA, vol.2 pp.914–918, (1993).

[188] Whitley, D.: *The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best* Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, California, USA: Morgan Kaufmann Publishers, pp.116–121, (1989).

[189] Whitley, D.; Starkweather, T.; Fuquay, D.: *Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator.* Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, CA, USA, (1989).

[190] Whitley, D.; Gordon, V.S.; Mathias, K. *Lamarckian Evolution, the Baldwin Effect and Function Optimization.* Proceedings of the third Conference on Parallel Problem Solving from Nature. Lecture Notes in Computer Science, vol.866, Springer, Berlin, pp.6–15, (1994).

[191] Yager, R.R.; Zadeh, L.A.: *Fuzzy Sets, Neural Networks and Soft Computing.* Van Nostrand Reinhold, New York, (1994).

[192] Yager, R.R.; Filev, D.P.: *Essentials of Fuzzy Modeling and Control.* ISBN: 0-471-01761-2, 408 pages, (1994).

[193] Yoon, H-S.; Moon, B-R.: *Synergy of Multiple Crossover Operators in a Genetic Algorithm.* Morgan Kaufmann (publisher), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Whitley, D.; Goldberg, D.; Cantu-Paz, E.; Spector, L.; Parmee, I.; Beyer, H-G. (editors), Las Vegas, Nevada, USA, (2000).

[194] Zadeh, L.A.: *The Concept of a Linguistic Variable and its Application to Approximate Reasoning.* Information Sciences - Application to Approximate Reasoning I - III, (1975).

[195] Zadeh, L.A.: *Knowledge representation in fuzzy logic.* In Yager, R.R. and Zadeh, L.A. (Eds.), An introduction to Fuzzy Logic Applications in Intelligent Systems, 1–25, Boston: Kluwer Academic, (1992).

[196] Zhang, J.; Renners, I.; Knoll, A.: *Genetic Adaptation to Optimal Membership Functions for Modelling with B-Splines.* 6th International Workshop Fuzzy-Neuro Systems, Leipzig, (1999).

Index