Philipps Universität Marburg

Fuzzy Operator Trees for Modeling Utility Functions

Dissertation

zur

Erlangung des Doktorgrades

der Naturwissenschaften

(Dr. rer. nat.)

dem
Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg
vorgelegt von

Yu Yi
aus Changsha, China

Marburg/Lahn 2008

Vom Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg
als Dissertation am 08.12.2008 angenommen.
Erstgutachter: Prof. Dr. Eyke Hüllermeier
Zweitgutachter: Prof. Dr. Bernd Freisleben
Tag der mündlichen Prüfung am 12.12.2008

Abstract

In this thesis, we propose a method for modeling utility (rating) functions based on a novel concept called **Fuzzy Operator Tree** (FOT for short). As the notion suggests, this method makes use of techniques from fuzzy set theory and implements a fuzzy rating function, that is, a utility function that maps to the unit interval, where 0 corresponds to the lowest and 1 to the highest evaluation. Even though the original motivation comes from quality control, FOTs are completely general and widely applicable.

Our approach allows a human expert to specify a model in the form of an FOT in a quite convenient and intuitive way. To this end, he simply has to split evaluation criteria into sub-criteria in a recursive manner, and to determine in which way these sub-criteria ought to be combined: conjunctively, disjunctively, or by means of an averaging operator. The result of this process is the qualitative structure of the model. A second step, then, it is to parameterize the model. To support or even free the expert form this step, we develop a method for calibrating the model on the basis of exemplary ratings, that is, in a purely data-driven way. This method, which makes use of optimization techniques from the field of evolutionary algorithms, constitutes the second major contribution of the thesis.

The third contribution of the thesis is a method for evaluating an FOT in a cost-efficient way. Roughly speaking, an FOT can be seen as an aggregation function that combines the evaluations of a number of basic criteria into an overall rating of an object. Essentially, the cost of computing this rating is hence given by sum of the evaluation costs of the basic criteria. In practice, however, the precise utility degree is often not needed. Instead, it is enough to know whether it lies above or below an important threshold value. In such cases, the evaluation process, understood as a sequential evaluation of basic criteria, can be stopped as soon as this question can be answered in a unique way. Of course, the (expected) number of basic criteria and, therefore, the (expected) evaluation cost will then strongly depend on the order of the evaluations, and this is what is optimized by the methods that we have developed.

Keywords : utility function, rating function, quality assessment, fuzzy set, fuzzy operator, evolution strategies, regression, ordinal classification, cost minimization.

Zusammenfassung

In dieser Arbeit stellen wir eine Methode vor, um Bewertungsfunktionen zu modellieren, die auf einem neuartigen Konzept der Fuzzy Operator Bäume (kurz **FOT**) basieren. Wie der Name andeutet, nutzt diese Methode die Techniken aus Fuzzy Set Theorie und implementiert eine Fuzzy Bewertungsfunktion, nämlich eine Funktion, die das Einheitsinterval abbildet, wobei 0 der niedrigsten und 1 der höchsten Bewertung entsprecht. Obwohl die erste Motivation von Qualitätsbewertung aus dem Bereich der Produktsteuerung kommt, ist unser Modell völlig generell und deshalb überall einsetzbar.

Unsere Methode macht es möglich, dass ein menschlicher Experte ein Model in Form eines FOT in einem sehr intuitiven und attraktiven Weg spezifiziert. Schließlich braucht er nur ein Hauptkriterium in mehrere Unterkriterien rekursiv zu zerlegen, und entscheidet, in welche Art und Weise die Unterkriterien zu kombinieren sind: Konjunktion, Disjunktion oder im Sinne eines Durchschnitt-Operators. Das Resultat ist die qualitative Struktur des FOT Models. In einem zweiten Schnitt wird dann das Model parametrisiert. Um den menschlichen Experten dabei zu unterstützen, oder ihn sogar abkömmlich zu machen, haben wir eine Methode zur Kalibrierung eines Models entwickelt, die auf exemplarischen Bewertungen basiert, in anderen Worten, rein daten-basiert ist. Diese Methode, die die Optimierungstechnik der Evolutionsstrategie verwendet, bildet den zweiten Hauptbeitrag dieser Arbeit.

Der dritte Hauptbeitrag dieser Arbeit ist eine Methode zur Evaluierung eines FOTs unter Berücksichtigung der Evaluierungskosten. Allgemein gesehen ist ein FOT eine Aggregation, die die Evaluierungen mehrerer fundamentaler Kriterien zu einer gesamt Bewertung eines Objektes kombiniert. Die Kosten für dieser gesamt Bewertung ist im Wesentlichen die Summe allen Evaluierungskosten der fundamentalen Kriterien. Aber, eine präzise Bewertung ist nicht immer notwendig, stattdessen, reicht es oft aus, in manchen Situationen sicherzustellen, dass die Bewertung über oder unter einem wichtigen Schwellenwert liegt. Darüber hinaus kann ein Evaluierungsprozess (sequentielle Evaluierung der fundamentalen Kriterien) gestoppt werden, so lange diese Frage eindeutig beantwortet werden kann. Natürlich sind die erwarteten Evaluierungskosten und Anzahl der fundamentalen Kriterien stark abhängig von der Ordnung der Evaluierung, die durch unsere neue Methode auch optimiert wird.

Schlüsselwörter: Bewertungsfunktion, Qualitätsmessung, Fuzzy Set, Fuzzy Operatoren, Gradientenabstieg, Evolutionsstrategie.

Contents

1 Introduction

In this chapter, we give an introduction to the main topic of this thesis, namely modeling utility functions using *fuzzy operator tree* (**FOT** for short). After the motivation in Section 1.1, we give the basic idea of FOT and the main techniques used in this thesis in Section 1.2. Finally we give an overview of the organization of this thesis in Section 1.3.

1.1 Motivation

The objective of this section is to introduce the main motivation of this thesis. We begin with the idea to apply automatic quality assessment instead of human experts. Then the problem by traditional quality assessment is demonstrated and summarized. To this end we generalize the expectations of a new method for modeling utility functions.

1.1.1 From Sensing to Automatic Quality Assessment

The original idea of modeling with FOT is motivated by quality assessment in product control, where the evaluation and assessment of products become a fundamental task. Due to the growing global competition, companies are continuously seeking for efficient and short developmental periods. At the same time, the demand on product quality and consequently customer satisfaction arises recently, so that systems to ensure products or services and assess quality of products have gained increasing importance. In this regard, quality control and quality engineering [Pyz03] has become an attractive research area in last decades. We will focus on one of its current research activities, namely automatic quality assessment (or quality evaluation), in this thesis.

Until now, many quality assessments are usually carried out by a human *expert* manually. By definition, an expert is a person who is well knowledgeable about the object, e.g. a food expert of quality control is a person who can identify quality of food well, etc. An expert is also called *decision maker* in the decision making problems [Saa94], since the knowledge of experts is expressed in the form of decisions in this case. This is not always a convenient way because of following difficulties:

1. First of all, there are usually only limited human resources available in practice, but many problems have to be solved. From financial aspect it is not favorable, because human experts are not always available and maintaining human experts is in most case more expensive than using measuring instruments.

2. The assessment is relatively subjective, since human experts may give different judgments even under identical conditions, and variable outcomes can be drawn by the same expert from time to time.

However in a globalized world, the objective, efficient quality assessment schema has played an essential role for quality control independent of place of production, especially when products or components of products come from different regions.

One trend to overcome the aforementioned difficulties is to apply electronic measuring instruments to take the place of human experts. For example, the company "Battenberg Robotic"[1] develops robotics as high-precision sensory measuring instruments (see Figure 1.1), which are specially adapted for constraint-dependent measuring tasks in automatic production and control processes for the purpose of guaranteeing an objective provable product assessment for producer and component supplier.

While the automatic acquisition of quality measurements has already been progressed in many industrial fields, the development of automatic quality assessment based on acquired measurements is still not satisfied, since the automatic quality evaluations are mostly strongly deflected from those empirical evaluations from human experts. We shall review traditional quality assessment and its disadvantages in next section.

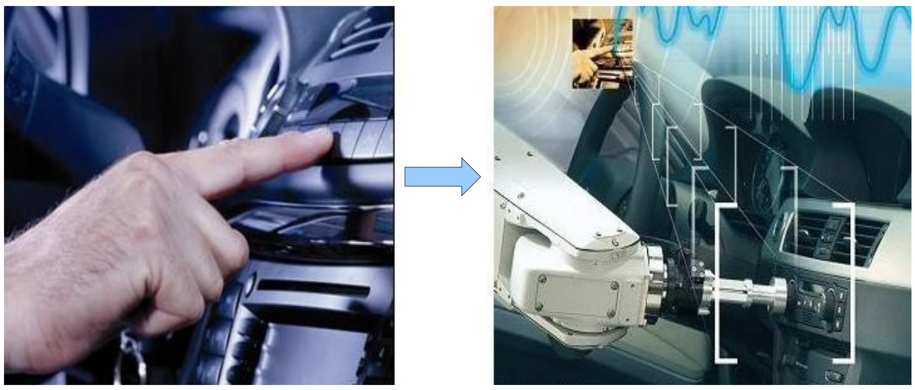


Figure 1.1: From sensing to automatic assessment

1.1.2 Traditional Quality Assessment and its Disadvantages

Traditional quality assessment usually relies on predefined "tolerance" domains, which indicate the desired intervals for measured criteria. A product is then labeled as "in order" (abbreviated, *IO*), if all measured criteria lie within given intervals, otherwise it is "not in order" (or "out of order", abbreviated *NIO*), no matter how many criterion failed.

Let's consider a concrete example, in which the goal is to evaluate a technical device such as the control panel of car radios, see Figure 1.2, where the quality depends on the operability of functional components: The switch button *A*, the adjusting knob *B* and several functional buttons *C*, *D*, *E*, and so on. To assess the quality of control panel automatically, a robot might be applied to test the operability of components, in which a robot pushes buttons to predefined pressure, and drives back afterward, so that the buttons return their starting positions. At the same time, several measurements are recorded, for example, the maximal position or strength of robot to reach predefined pressure.

For adjusting knobs like *B* here, a robot rotates the adjusting knob into a predefined position, and again rotates it back into the starting position, at the same time similar measurements are recorded. For the sake of clearness, let us assume the operability

[1] www.battenberg.biz

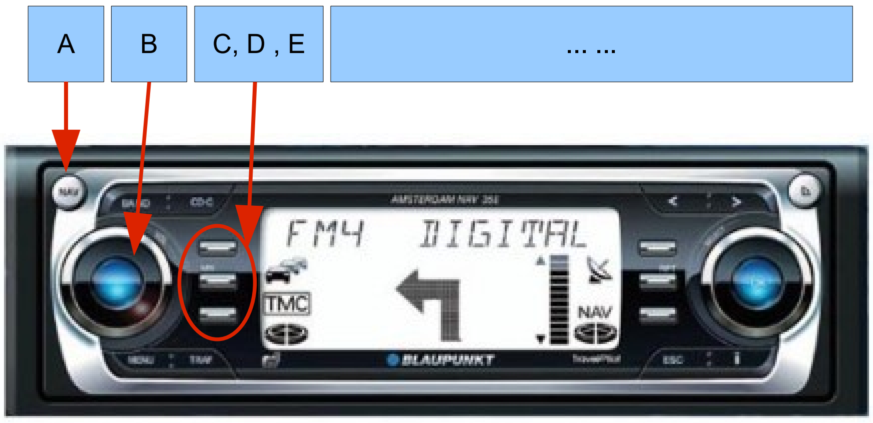


Figure 1.2: Quality evaluation of car radio

of components depends only on the maximal strength (unit: N) of the robot to reach predefined pressure or positions (*maximal strength* for short) in this example.

According to a human expert, the quality of car radio can be assessed under following principles:

- A car radio is “good” operable (IO), if the operabilities of its components are not too “far” away from (standard) desired values;
- A car radio is “good” operable (IO), if all its components can be operated in almost “similar” way, for example, the maximal strength on components with similar functionalities (e.g. C, D and E here) should be the same approximately, or the proportion of maximal strength of components (e.g. A and B) should be relatively constant;
- Otherwise a car radio is “bad” operable (NIO).

In traditional quality assessment, the involved criteria would be tested sequentially, and the result is a set of measurements, say $x_1, \ldots, x_n$. For each measurement x_i, there is a tolerance interval $[l_i, u_i]$, where l_i (u_i) indicates the lower (upper) bound in which a measurement is IO. Then the device is IO if $x_i \in [l_i, u_i]$ for all $i \in \{1, \ldots, n\}$, otherwise it is NIO.

Table 1.1 lists three car radio examples with measurements of maximal strength at five tagged control points in Figure 1.2 respectively, where we assume that the tolerance domains for these measurements are given. According to traditional quality assessment, these samples are classified into IO and NIO categories, as the last column in Table 1.1 indicates.

Table 1.1: Car radio examples with maximal strength (unit: N) at five control points and overall quality using traditional quality assessment

	A	B	C	D	E	Quality
$[l_i, u_i]$	$[0.5, 0.9]$	$[0.2, 0.4]$	$[0.4, 0.8]$			
Nr. 1	0.7	0.3	0.5	0.4	0.7	IO
Nr. 2	0.49	0.3	0.6	0.6	0.6	NIO
Nr. 3	0.9	0.41	0.8	0.8	0.8	NIO

However the quality assessments in Table 1.1 are problematic from a human expert’s point of view, the first sample is labeled as IO, though quite different values on last three buttons (C, D and E) are measured, which have similar functionalities. On the other hand, the as NIO labeled second sample comes with “perfect” buttons except A, whose value lies just a little outside the tolerance domain. Regarding this, one might question whether the second sample is still more preferable for users than the first one. For the third sample, all buttons seem more cumbersome than the standard settings, since the values all reached or exceeded the upper bound of related intervals a little. For a human expert, the third sample may still be better than the first one.

Traditional quality assessment, which is common used in industrial quality control, exhibits a number of obvious disadvantages:

1. Firstly, traditional quality assessment is very coarse, as it only distinguishes between products that are *IO* and *NIO* two categories, which is usually not sufficient in many applications, where various scales of quality assessment are desired. Needless to say, a more refined, intelligent quality assessment will often be desired in practice, which can provide discrimination between different quality levels. For instance, following additional terms are often used concerning quality assessment in practice: “middle”, “good with few flaws” or “superior” etc, which can be very useful in that, for example, products with “middle” quality can still be accepted under price reduction, instead be labeled as *NIO* simply, or products can be assembled according to differentiated quality classes, etc.

2. Secondly, traditional quality assessment usually evaluates each measured criterion separately by comparing with respective standard values and allowed deviations (expressed in the form of intervals) simply, however, a human expert is used to consider an evaluated product as a whole, instead of independent criteria[2]. While traditional quality assessment works in a sequential way by checking measured criteria one by one, for a human expert it is important that an overall impression on evaluated product determines its overall quality, so several “bad” criteria can be compensated by other “superior” criteria. Let us call this property as *compensation*. From the logical point of view, traditional quality assessment bases on purely conjunctive combination of single criteria, which does not allow for any kind of compensation, as a result, even a large number of almost perfect values cannot compensate for a single critical measurement. The consequence of such assessment procedure is also very sensitive towards measurement errors, so in practice, the reject rate is often extremely high, since the probability to have always a “good” measurement ($x_i \in [l_i, u_i]$, for all $i \in \{1, \ldots, n\}$) will almost vanish for a large enough number of measurements.

   Using tolerance intervals causes the well-known threshold effect, namely a minimal change of a single measurement around boundaries of tolerance intervals may lead to a completely different evaluation, which is of course undesirable in practice. A good example can be found in Table 1.1, where the second example is labeled as *NIO* just because of the maximal strength upon A lies a little under the tolerance domain, which might be error in measurement.

1.1.3 Problem of Modeling Utility Function

In this thesis, we want to develop a new method for modeling utility function (or rating function), which can be used to assess any sort of object, entity or alternative, for example products in the previous case. A utility function is able to assign a numeric degree (called utility or rating) for alternative in place of human experts, which can be seen as satisfaction or recommendation on related alternative. Note

[2] Actually the relationship between criteria could still be specified as additional criteria in traditional quality assessment. For instance, to express that button C and D should be equally operable, one can add a new measurement to compute difference between maximal strength on C and D. However a flexible compensation among criteria is by no means a trivial task in traditional quality assessment.

that a utility function can be applied in several tasks, for instance, to select the best alternative, make a ranking of these alternatives or a classification, etc.

Amongst other, the motivation of this thesis is to develop a new method for automatic quality assessment, which concerns following aspects:

- **Compensation**: It ought to mimic human experts by automatic quality assessment as accurate and intelligent as possible. In contrast to traditional quality assessment, the new method should allow compensation of evaluated objects, just like human experts usually do;
- **Gradual quality scales**: Instead of simple "*IO-NIO*" categories in traditional quality assessment, it is proposed that the new method is able to support a more refined discrimination between quality levels of objects, in which traditional quality assessment can be viewed as a special case with only two quality levels. In the new method, the stability of automatic quality assessment should be improved. That is, random deviation or error in measurement should not affect the overall quality dramatically. The threshold effect should be avoided in the new method.

1.2 Modeling with Fuzzy Operator Tree

In this thesis, we propose a new hierarchical model for quality assessment to overcome those disadvantages caused by traditional quality assessment, subsequently referred to as *fuzzy operator tree* (FOT), which makes use of tools and techniques from fuzzy set theory by recursively decomposing a utility criterion into sub-criteria. A fuzzy operator tree implements a *fuzzy rating function*, that is, a rating function that maps to the unit interval $[0,1]$, where 0 corresponds to the lowest and 1 to the highest evaluation. Even though the original motivation for developing an FOT model comes from quality control, we like to emphasize that FOTs are much more general, due to the fact that an FOT is simply a special kind of rating function, and can in principle be used for rating all sorts of things. This is why we shall often use a neutral term like "alternative" instead of a more special one such as "product".

In following text, we give a simple example of FOTs in Subsection 1.2.1 and discuss the advantages of using FOTs in Subsection 1.2.2. The mainly applied techniques are introduced in Subsection 1.2.3. Finally the challenges to build an FOT are listed in Subsection 1.2.4.

1.2.1 A Simple Example of Fuzzy Operator Tree

Figure 1.3 shows a simple graphical illustration of an FOT, where the assessment of candidates for a certain job is decomposed into two criteria *formal* skills and *language* skills. The former criterion is further decomposed into skills in mathematics (*math*) and computer science (*CS*), and the latter into *French* and *Spanish*.

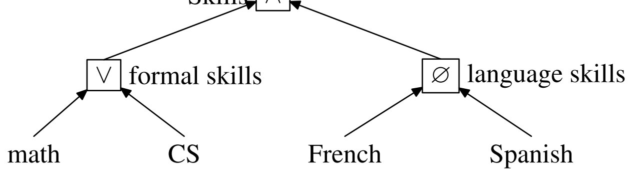


Figure 1.3: FOT example for the assessment of candidate

Note that the $\wedge$, $\vee$ and $\varnothing$ stand for parameterized fuzzy aggregation operators *AND*, *OR* or *Average* respectively. At the lowest level of an FOT, the inputs of an FOT are basic evaluations on different criteria, for example here in this example, *math* denotes a basic criterion upon the mathematic knowledge of candidate. Regarding this, an input of the FOT might be a measurement related the mathematic knowledge of candidate, for instance, a degree in a mathematical examination, which is modulated by an associated fuzzy set.

This example can be interpreted as following: The skills of a candidate depends on formal and language skills, which are aggregated using an *AND* operator, where formal skills depend on both mathematics and computer science, but they are combined with an *OR* fuzzy aggregation operator, while the language skills are supposed to depend on French and Spanish, which are combined with a fuzzy aggregation operator *Average*. The inputs of this example are measurements related to four basic criteria (math, CS, French and Spanish), which might be simply degrees derived from transcript, or modified according to the particular interest of the decision maker. The input of this example would firstly be modulated by several fuzzy sets, through which several numeric degrees in unit interval $[0,1]$ are deduced. These numeric degrees are typically preference or satisfaction degrees with respect to a given criterion, where 1 expresses perfect satisfaction on a related criterion and 0 corresponds to totally unacceptable, they are called "basic evaluations" in this thesis. A scale between 0 and 1 should demonstrate how far a criterion is satisfied from the prospect of a decision maker. One can read the output (also called *utility*) of an FOT from the root node, here "Skills", where a numeric scale would be returned also in a unit interval $[0,1]$, a preference degree tells how a decision maker favorites an alternative represented by the mentioned criteria.

To apply this example, the related measurements according to an alternative are put into an FOT at the leaf nodes. The evaluation of this alternative can be read from the root node of the FOT, in this way one can compare a set of alternatives, or obtain the quality of single alternative.

1.2.2 Features of Fuzzy Operator Trees

Amongst other, an FOT is characterized by the following properties:

1. **Modulation of measurements**: At the lowest level, any measurement, say x_i, is modulated by an associated fuzzy set, for example, if x_i has the definition domain $\mathbb{R}$, then $f_i : \mathbb{R} \to [0,1]$. As a result, a measurement indicates a basic evaluation to the overall quality. The evaluation $(f_i(x_i))$ is a number in the unit interval, expressing to what extent x_i meets the requirement from the human expert's point of view. As generally known, applying fuzzy sets provides a gradual transition between satisfaction and violation of the range constraint, and is able to distinguish more carefully between weak and strong violations of the range constraint, therefore to avoid the above-mentioned threshold effect of traditional quality assessment and improve the stability of utility function. Implicitly, a membership degree $f_i(x_i)$ provides information about how far away x_i is from an ideal value. On the other side, the basic evaluation modulated by fuzzy sets can be interpreted in the natural language intuitively, therefore can be well understood by human beings.

   As another potential advantage of the above approach, we note that it allows one to treat different types of measurements in a unified way. In fact, using fuzzy sets to modulate a single measurement variable is only a special case, in which it is possible to express restrictions on the relation between several measurements in principle, which is especially useful to formalize (fuzzy) constraints on more than one measurement variables. For example, in order to express the constraints in the form of "x_i and x_j are almost equal", where x_i and x_j are two measurements, a fuzzy set based on x_i and x_j can be defined, and the membership degree $f_i(x_i, x_j)$ is a number in the unit interval that has a proper semantics and can be compared and combined with other values.

2. **Hierarchical structuring of a utility function**: Constructing an FOT follows the criterion-decomposing manner, namely a branch in FOT can be interpreted as decomposing a criterion into several sub-criteria, this is an intuitively appealing and commonly used strategy, so that an FOT built in this way is always comprehensible and can be accepted and understood well by human beings.

   The hierarchical, modular structure of an FOT allows one to specify utility functions in a very systematic way, which is of special importance for assessing the quality of technical products. Typically, a technical product itself has a modular structure and therefore can be decomposed into subcomponents in a recursive way. The "divide-and-conquer" strategy underlying FOTs makes the assessment of very complex systems controllable.

   Notice that the output of an FOT is a numeric scale in unit interval $[0,1]$, as well as the intermediate results and basic criteria of FOT, which can be easily understood and used in practice. The nearer a scale to 1, the better quality an evaluated object has. Reversely, the nearer a scale to 0, the worse quality an evaluated object has.

3. **Parameterized fuzzy aggregation operators** : To aggregate the evaluation of sub-criteria, say C_1 and C_2, into an evaluation of a criterion C, we make use of three types of operators that support different combination modes:

   - Conjunctive combination: If the satisfaction of C requires the satisfaction of both C_1 and C_2, a triangular norm (t-norm) is used as an aggregation function [KMP02];
   - Disjunctive combination: If the satisfaction of C only requires the satisfaction of C_1 or C_2, a triangular conorm (t-conorm) is used as an aggregation function [DP80];
   - Averaging: If the evaluation of C is supposed to be an average between the evaluations of C_1 and C_2, we make use of an OWA (ordered weighted average) operator [Yag88].

   Since all three types of operators are associative, the number of sub-criteria to be combined can of course be larger than two.

   A conjunctive combination is the most stringent type of aggregation, while a disjunctive combination is the least stringent one. Within these two classes of operators, it is possible to make an even finer differentiation by looking at the order relation between t-norms (t-conorms) [DP80]. Our solution here is to use parameterized families of t-norms (t-conorms), for example, the well-known Hamacher family [Ham78], most of which include the commonly used operators as special cases.

   On the other side, the class of OWA operators nicely "fills" the gap between the largest conjunctive combination (minimum) and the smallest disjunction combination (maximum), we thus obtain a continuous spectrum of aggregation operators.

4. **Weighting sub-criteria** : In FOTs it is desirable to weigh one sub-criterion higher than another one, which is problematic for t-norms and t-conorms since these are symmetric fuzzy operators by their very nature. We make use of

a simpler alternative that resorts to the idea of *linguistic modifiers* [Zad72, Lak73]. A linguistic modifier is a function $m : [0,1] \to [0,1]$ that depicts the effect of linguistic hedges. Note that in the previous example of FOTs, the use of linguistic hedges is omitted for the sake of clearness.

5. **Interpretability** : In contrast with other utility function modeling methods [Bly02, Saa94, Dye90, HHR$^+$03, HH03, HS02, FM03a], FOT especially enjoys its excellent interpretability. The interpretability denotes the capability to express the behavior of the real system in an understandable way. The interpretability of an FOT is guaranteed by using tools and techniques from fuzzy set theory, namely:
   - Firstly the use of fuzzy sets for modulation of measurements has good interpretability, since, as is well known, the fuzzy sets can be interpreted in the form of natural language. Linguistic terms with vague meaning, which are often used by human beings, can be well expressed in terms of fuzzy sets, such as: "old", "not far away from" and "approximately equal" etc.
   - Secondly the fuzzy aggregation operators used in FOTs not only provide great flexibility to combine sub-criteria in various way, but also reserve the meaningful interpretations "AND", "OR" or "Average". Although parameter specification can change the behavior of aggregation operators, their linguistic interpretations are kept.
   - Thirdly the linguistic modifiers to represent the weight of sub-criteria have also good interpretability.

Using an FOT to model the quality assessment process can fulfill the aforementioned expectations on automatic quality assessment completely, namely:

1. FOTs employ fuzzy sets to modulate measurements, and fuzzy aggregation operators to combine sub-criteria, which makes it possible to formulize the constraints on different criteria easily and compensate different criteria in a more flexible way. Since the compensation of criteria can be modulated in FOTs, using an FOT enables a much more accurate modeling of human experts than traditional quality assessment.

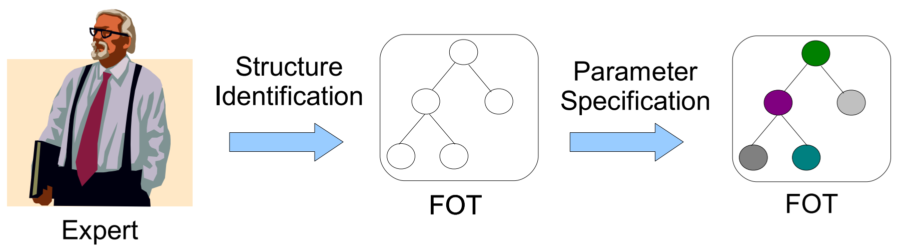
2. The numeric output of FOTs provides more flexibilities than traditional quality assessment, and can support a more refined discrimination between quality levels. For example, one can convert a numeric scale into "*IO-NIO*" decision by employing a single threshold, say α in $[0,1]$, which divides the unit interval into two regions ($[0,\alpha)$ and $[\alpha,1]$). Beside this, the numeric scale is more flexible than simple decision, which is especial useful for other tasks, like ranking, comparison, etc. The stability of automatic quality assessment is improved strongly by applying fuzzy sets and fuzzy aggregation operators in FOTs, the threshold effect can be avoided completely.

Although the original motivation for this thesis comes from the field of quality control, FOTs provide a completely general way for modeling utility functions. Moreover, as will be discussed in later parts of the thesis, FOTs can be applied in several other different tasks, for example classification problems and information retrieval in machine learning [HMS02, Mit97, Yag00], or in new researching areas like preference modeling [CP04, PR02] or label ranking [BH07]. Generally speaking, FOTs provide powerful tools for assessment of any sort of object, entity, or alternative. In this regard, we employ the abstract concept of *utility*, mathematically formalized in terms of a *utility function*, instead of rating or quality, which has a longstanding tradition in economics field, where it plays an important role in the study of economic behavior [GS88].

1.2.3 How to Build a Fuzzy Operator Tree

Building an FOT consists of two steps: Structure identification and parameter specification, where the former step aims to allow a human expert to specify a model in the form of an FOT in a quite convenient and intuitive way. To this end, the latter step specifies the parameters involved in FOTs. Figure 1.4 demonstrates the process to build an FOT, where we use different colors to emphasize the parameter specification for fuzzy components in an FOT. We give more details about these two steps in following text.

Figure 1.4: Modeling utility function with FOT



1.2.3.1 Structure Identification

In this step we want to extract the abstract structure information of an FOT with the help of a human expert, which is based on three important conceptions, namely:

- Modulation of measurements in terms of fuzzy sets, in which the basic evaluation of involved criteria as inputs at the lowest level of the FOT are modulated in terms of associated fuzzy sets. According to special interest of human expert on alternatives, this phase serves to extract the (fuzzy) constraints on measurements in the form of fuzzy sets. To this end, different types of measurements are converted in a unified way, namely as fuzzy membership degrees.
- Hierarchical structuring of a utility function, in which we intent to elicit how human experts assess the quality of evaluated alternatives. Human experts are allowed to decompose a criterion into sub-criteria in a recursive way, until all criteria are modulated in previous phase already. Following this idea, the hierarchical structure of an FOT can be built in an intuitive, easy way.
- Flexible aggregation of sub-criteria by means of parameterized fuzzy aggregation operators, in which human experts are required to determine the type of decomposition involved in an FOT by means of combination “AND”, “OR” or “Average”, these combinations can be expressed in the form of a parameterized t-norm, t-conorm or OWA operator.

1.2.3.2 Parameter Specification

In this step, we concentrate on the specification of the parameters of the aggregation operators, linguistic hedges as well as the fuzzy sets. Sometimes this task can also be done by human experts directly, for example, in quality assessment of technical products, where the parameterization is usually predefined according respective standards. But in other situations, the expert may have problems with specifying precise parameters but only the abstract structure.

In this thesis, we concentrate on a special alternative to support a human expert in designing FOTs, namely learning parameter from data. The main task here is addressed as: Given the structure of FOT and a set of exemplary data provided by human experts, we intend to find a parameter specification, which can mimic the behavior of human experts as well as possible, the quality of a parameter specification depends on the agreement between evaluations given by human experts and predicted by an FOT under a given specification. A calibration method is developed in this thesis for fitting the parameters in FOTs. As the task in this step becomes a typical learning problem, several techniques are used in this thesis for purpose of parameter specification: Gradient descent [Mit97], simulated annealing [KGV83, SPR04] and evolutionary algorithms [Wei02, Coe06]. In following text, we use the term *Calibration* instead of specification of parameters, in the sense that the structure of an FOT would be kept during this step.

1.2.4 Challenges

Modeling with FOTs encounter following challenges:

- Knowledge elicitation: How can we extract the prior knowledge of a human expert with a loss of information as small as possible? Many current techniques intend to deal with knowledge elicitation with different types of models. Our starting point is based on the assumption that a utility function can be expressed in terms of a hierarchical structure by decomposing criteria into sub-criteria in a recursive way. The evaluations of sub-criteria can be combined by means of aggregation operators of different character: Conjunctive, disjunctive and averaging.
- Calibration: As a special learning problem with given exemplary data, how can we specify the parameters in FOTs to mimic human experts as accurate as possible?
- Minimizing evaluation cost: After building FOTs through structure and system identification processes, an issue of minimizing evaluation cost is addressed by applying FOTs, since firstly a precise utility is not always necessary in practice, for example, in many cases one prefers ordinal utility rather than numeric one. Secondly the evaluations of criteria involved in FOTs have usually different evaluation costs. So from the economical point of view, an order of to be evaluated criteria is required, so that the overall evaluation cost could be minimized.

1.3 Outline of the Thesis

The main topic of the previous section are also reflected in the organization of this thesis.

- Chapter 2 introduces the methods of fuzzy modeling. At first we give the basic components of fuzzy set theory and their properties. Then we present a framework to combine them together, which ought to be a powerful tool to modulate utility functions.

- Chapter 3 discusses the problem of calibration of FOTs, where this problem is defined formally, then several techniques for the purpose of calibration are introduced with respective experiments.
- Chapter 4 discusses the problem of minimizing evaluation cost and corresponding approaches to arrange the evaluations on FOTs efficiently.
- Chapter 5 discusses related work. We shall review several similar works and explain the differences to FOT modeling.
- Chapter 6 concludes this thesis with a summary of its results. Additionally, we outline future research directions from presented conclusions.

Bibliographic note : The main contributions of Chapter 2 and 3 have been published in [YHF08]. Chapter 2 is partly based on [Zae05].

2 Modeling Utility Functions with Fuzzy Operator Trees

This chapter is devoted to the basic conceptions and techniques for modeling utility functions with FOTs. To this end, in order to build an FOT model, a human expert just needs to simply split evaluation criteria into sub-criteria in a recursive manner and to determine in which way these sub-criteria ought to be combined.

In the following text, we give the details on how to model utility function in the form of FOT with the help of human experts. In Section 2.1 we define the problem of modeling utility functions and introduce some related terminologies. After the components of fuzzy set theory for constructing an FOT are described in Section 2.2, we present the fuzzy operator tree and its properties in Section 2.3. The elicitation methods for modeling FOTs are shown in Section 2.4. Section 2.5 concludes this chapter.

2.1 Preliminaries

Let us start with the formal description of modeling utility functions, then we introduce the related terminologies for modeling utility functions.

2.1.1 Utility and Utility Functions

In economics, *utility* (also called *rating*) is a measure of the relative satisfaction or desiredness of the consumption of goods [NM53]. To represent utilities (ratings), it is convenient to apply utility functions, then the consumption of goods can be compared in a unified form. The consumer's utility function u is defined as:

$$u : X \to \mathbb{R}$$

namely a mapping from a consumption good set X to the real number space $\mathbb{R}$. For any $x, y \in X$, $u(x) \geq u(y)$ indicates the customer strictly prefers x to y or there is no difference between x and y from the customer's point of view.

Borrowed from economics, the abstract concept of utility and utility function respectively plays a fundamental role for the evaluation and assessment of any sort of object, entity or alternative involved in numerous types of applications [GS88]. In decision making, for instance, where a final choice has to be selected among alternatives, the preference of available alternatives is presented in the form of utilities deduced by some kind of utility function. In different context, the utility is also named as preference, rating, or quality [CP04].

Let us consider the problem to find an apartment as a concrete example using utility and utility function: Two attributes are considered here, *distance D* and *size S*. The distance is calculated in miles from the apartment to the working place, and we assume that it ranges over the interval $[0, 100]$, the apartments lying more than 100 miles away from the working place are treated as unacceptable. The size of an apartment is measured in squared meters, and ranges over the interval $[20, 100]$. Given a set of available alternatives (listed in Table 2.1), the goal of decision making is to determine the "best" choice according to the preference of the decision maker. Suppose that a utility function in this case can be expressed as a linear function:

$$u(d,s) = \begin{cases} 1.2 \cdot (1 - \frac{d}{120}) + 0.7 \cdot (\frac{s}{100}) & \text{if } d \in [0,100] \text{ and } s \in [20,100] \\ 0 & \text{otherwise} \end{cases}$$

one can calculate the utilities for given alternatives, and make a decision in favor of the offer A.

Table 2.1: Example of decision making using utility function

Alternative	*A*	*B*	*C*	*D*
Distance	2.4	40	31	85
Size	48	78	64	100
Utility	1.51	1.35	1.34	0.05

2.1.2 Modeling Utility Functions

Given a utility function, the evaluation process for alternatives is relative straightforward, in which one simply deliver all alternatives into utility function as input to get their utilities. Based on these utilities, a decision or ranking among alternatives can be determined by making a comparison on their utilities. To construct a "good" utility function is, however, the main challenge, because there exist numerous representations of utility functions, for example, parameterized linear functions, nonlinear (polynomial) functions, or functions in tabular form, and so on. Usually it depends on how the overall utility is related to representative features and how the features interact. Generally, there is no universal representation for all kind of problems. Even for a particular kind of representations, there exist still numerous variants. For instance, one can select different combinations of criteria involved in the utility functions, or different parameters. The process for determining a utility function is called *Modeling utility function*, which is also the main task for this chapter.

Formally, the problem of modeling utility function is defined as follows: Given a decision maker (e.g. human expert, customer), whose preference should be modeled using a utility function, and a set of objects or alternatives O, the goal of modeling utility function is to determine a utility function $u : O \to \mathbb{R}$, which reflects the decision maker's preferences on an alternative from O, that is, a function u, which assigns a real value for each alternative to express the preference of the decision maker, it holds:

$$\forall x, y \in O : u(x) > u(y) \Rightarrow \text{ the decision maker prefers } x \text{ to } y$$

Due to the variety of utility functions, there are two main criteria applied to describe the quality of a utility function:

- **Accuracy**
  The accuracy is one of most important criteria concerning the quality of a util-

ity function, which indicates how well a decision maker agrees to a corresponding utility function. There are a lot of common used strategies to measure the accuracy of utility functions, for example, given a set of alternatives, whose utilities are given by human expert, accuracy can be defined as the rate of correct predictions made by utility functions.

- **Interpretability**
  The interpretability denotes the capability of a utility function to express the preferences of a decision maker in an understandable way. Whereas the accuracy describes how well a utility function mimics a decision maker, the interpretability indicates how far a human being understands a utility function. The term of interpretability is replaced with simplicity in many works, since there the interpretability is defined as the abstract size of a utility function. Generally, the simpler a utility function is, the better one can understand it. The interpretability can be defined as, for instance, the number of linear functions, or parameters involved in an utility function, etc.

In this thesis, we specially restrict to express the utility in the unit interval $[0,1]$ instead of the real number space $\mathbb{R}$, that is, a utility function represented by FOT can be defined as:

$$u : O \to [0,1]$$

with O is domain of alternatives to be evaluated, where $u(o) = 1$ ($u(o) = 0$) denotes an alternative o has "best" ("worst") quality, generally the nearer the value $u(o)$ to 1, the higher quality of o, vice versa.

2.2 Fuzzy Set Theory

The notion of a fuzzy set stems from the observation made by L.A. Zadeh [Zad65], which emphasizes the gab between mental representations of reality and usual mathematical representations. Fuzzy set theory bases on binary logic, precise numbers, differential equations and so on, and is characterized such that it permits the gradual assessment of elements with the aid of the membership functions. As a mathematical tool fuzzy set theory is particularly desired for handling incomplete information, the unsharpness of classes of objects and situations. It offers a unifying framework for modeling various types of information ranging from numerical, to symbolic and linguistic knowledge [FY02].

In this section, we take benefit of fuzzy set theory to represent the knowledge of experts in the form of natural language. We introduce several main components of fuzzy set theory, which are employed to construct an FOT for modeling utility function, namely fuzzy sets, fuzzy aggregation operators and linguistic hedges. These three components serve to interpret different natural language terms, for example, the knowledge from an expert might be expressed as: "The quality of an auto depends on whether it has a very good engine and a low fuel consumption", where

- the basic properties of measurements, here "good engine", "low fuel consumption" and "high quality" would be modulated using fuzzy sets, as Subsection 2.2.1 describes. Whether an auto has these fuzzy properties, or belongs to these fuzzy sets, is a matter of degree.

- the connection terms, like "and" here, are accommodated by fuzzy aggregation operators, which give a meaningful connection between fuzzy membership degrees. We introduce the fuzzy aggregation operators in Subsection 2.2.2.

- the adverbs and adjectives of natural languages are interpreted with the aid of linguistic hedges, such as "very", "slightly" and "approximately", which we describe in Subsection 2.2.3.

2.2.1 Fuzzy Sets

As an extension of the classical notion of set theory, fuzzy set theory has been firstly developed as an useful concept for dealing with real world phenomena. In classical set theory, an element either belongs or does not belong to the set, for example, the class of animals clearly includes dogs, horses, birds, etc, and on the other hand excludes objects such as computers, desks, etc. Generally, in classical set theory the membership of the elements in a set is assessed in binary terms $\{0,1\}$, therefore, we will assign the membership 1 for dogs to the class of animals, and 0 for computers, since apparently, dogs are animals, whereas computers are not. However, the classical set theory encounters difficulties to describe objects with ambiguous status, like for bacteria it is hard to say whether it is an animal or not. More counterexamples come from the natural language of human beings, terms such as "old man", "small number", or "bald head" can not be precisely defined in the usual mathematical sense.

In contrast to the classical set theory, fuzzy set theory permits the gradual assessment of elements to a set. In this subsection, a fuzzy set is employed to represent the basic properties in the form of natural language, which is achieved with the aid of a membership function valued in the unit interval $[0,1]$. In fuzzy set theory, a membership function plays a basic and significant role, since it characterizes a fuzzy set and all treatments of those fuzzy sets made in terms of theirs membership functions. To present a fuzzy property like "low fuel consumption", a fuzzy set is created to describe how far that an auto has low fuel consumption from an expert's point of view.

2.2.1.1 Definition of Fuzzy Set and Membership Function

Definition 2.1 (Fuzzy set)**.** Let X be a domain of objects, a fuzzy set A over X is characterized by a membership function $\mu_A : X \rightarrow [0,1]$, which associates each object x in X with a real number in the interval $[0,1]$. $\mu_A(x)$ represents the membership of x to fuzzy set A. Usually the membership function $\mu_A(x)$ is abbreviated to $A(x)$.

Note that fuzzy sets generalize classical sets, because classical sets are just special cases of fuzzy sets, in which the membership function becomes the indicator function defined in following manner:

$$A(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases}$$

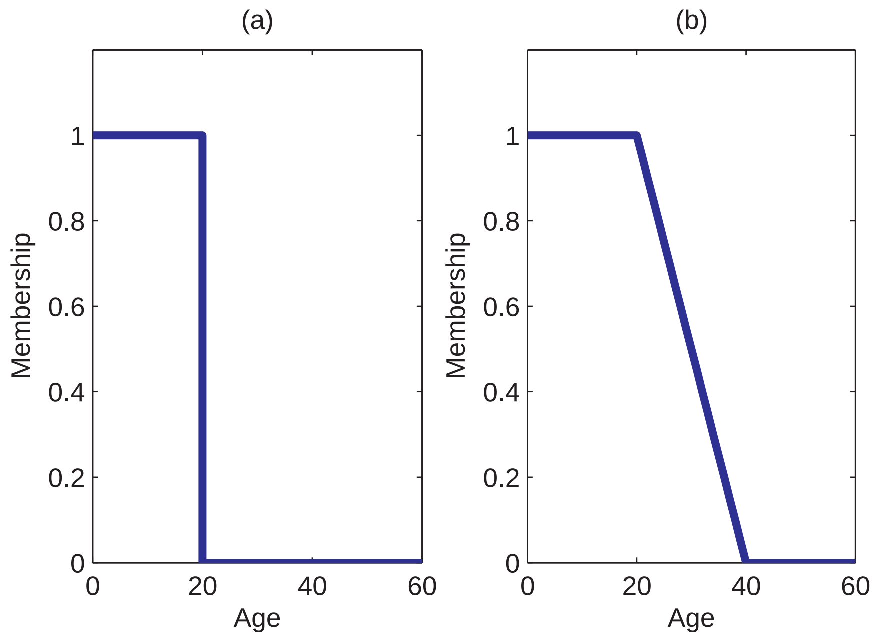
where x is an object instance and A a fuzzy set. Regarding this, classical sets are also called *crisp sets*, or Boolean sets. The fuzzy set serves as an expression tool for properties of objects in the form of "an object X is ...", for instance, "X is the capital of Germany" describes the property of a city where the government of Germany is carried out, whereas "X is old man" expresses the property of age of a human being according to the cognition about "old".

Let us consider an example with a fuzzy set A of "young man" (for human beings in a given context), which is based on a single variable "age". Traditionally, the classical set theory usually applies a single threshold, in which an arbitrary cut point is selected to help distinguishing whether a given person (with known age) is young or not. Suppose that a particular threshold is chosen, say 20 years, so that persons, who are exactly 20 years old or younger, are classified to belong to A. The corresponding membership function under classical set theory is plotted in (a) of Figure 2.1. Obviously, for a human being it is hardly to agree the statement that a young man suddenly becomes “not young” after his twentieth birthday. Fuzzy set theory solves this problem by setting a transition area between “young" and “not young", and assigning related objects with gradual memberships, for instance, the previous fuzzy set A can be defined as follows:

$$A(x) = \begin{cases} 1, & \text{if } x \leq 20 \\ \frac{|x-40|}{20}, & \text{if } 20 < x \leq 40 \\ 0, & \text{otherwise} \end{cases}$$

where x denotes the age, as in (b) of Figure 2.1 shows.

Figure 2.1: Membership functions for “young”: Classical set theory vs. fuzzy set theory



Based on fuzzy sets, several definitions are extended from classical set theory:

- A fuzzy set A is *empty*, if and only its membership function is identically zero on X. That is, $\forall x \in X : A(x) = 0$.
- For two fuzzy sets A and B, we say $A = B$ (A and B are *equal*), if and only if it holds $\forall x \in X : A(x) = B(x)$.
- The *union* of two fuzzy sets A and B is a fuzzy set C, written as $C = A \cup B$, whose membership function is related to those of A and B by

$$\forall x \in X : C(x) = S(A(x), B(x))$$

  where S is a t-conorm (see Section 2.2.2), a common used t-conorm is the max function.
- The *intersection* of two fuzzy sets A and B is a fuzzy set C, written as $C = A \cap B$, whose membership function is related to those of A and B by

$$\forall x \in X : C(x) = T(A(x), B(x))$$

  where T is a t-norm (see Section 2.2.2), a common used t-norm is the min function.

- The *complement* (or *negation*)of a fuzzy set A is denoted by A' and defined by

  $$\forall x \in X : A'(x) = N(A(x))$$

  where N is a negation operator, a common used negation operator is $N(x) := 1 - x$.

- The notion of *containment* from classical set theory is newly defined as: A fuzzy set A is contained in fuzzy set B (equivalently A is a subset of B, or A is smaller than or equal to B), if and only if $\forall x \in X : A(x) \leq B(x)$.

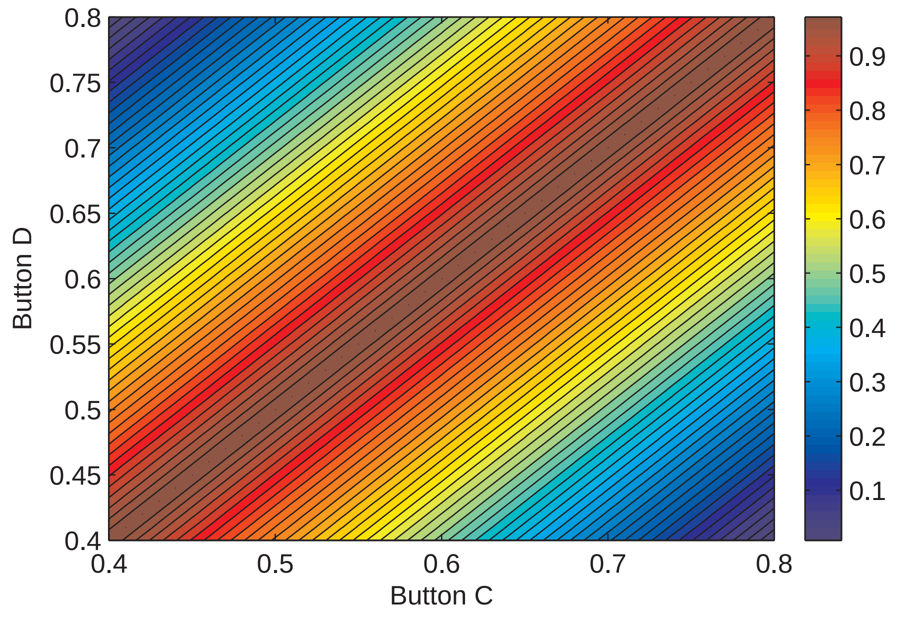
Generally, if we consider the dimensionality of X, a fuzzy set A over X is a mapping:

$$A : X_1 \times X_2 \times \ldots \times X_n \to [0,1]$$

where $X = X_1 \times X_2 \times \ldots \times X_n$ is the Cartesian product of n domains of variables $x_1, x_2, \ldots, x_n$. The previous example shows a fuzzy set of the dimensionality 1. A *fuzzy relation* R on $X_1 \times X_2 \times \ldots \times X_n$ is a fuzzy set defined on $X_1 \times X_2 \times \ldots \times X_n$ [Ovc02]. In the particular case when $n = 2$, a fuzzy set on the Cartesian product $X \times Y$ is a *fuzzy binary relation* .

For example, regarding the car radio example in Figure 1.2, let us suppose two continuous domains $X = Y = [0.4, 0.8]$ for the maximal strength of two buttons (C, D), a fuzzy relation A on $X \times Y$ may be used to express a property "Button C and D are equally operable", and defined by a two-dimensional membership function in Figure 2.2:

Figure 2.2: Fuzzy relation of "Button C and D are equally operable"



2.2.1.2 Types of Fuzzy Sets and Elicitation of Membership Functions

Note that a fuzzy set is a gradual representation of objects, which depends on the context of the applied area. For instance, a fuzzy set "high" should have different meaning for human beings and trees. Even for human beings, a fuzzy set can also have various expressions, like people do have different definitions of "high" from region to region. To define a fuzzy set and its membership function respectively, one direct method is to elicit the memberships representatively through direct communication with the knowledge source, namely experts [NK02]. Because a fuzzy set is identified through its membership function, to elicit a fuzzy set we just need to determine the memberships of potential objects related to this fuzzy set, such as to clarify a fuzzy set "tall man", the expert is required to give the "real" memberships for a provided set of finite participants, according to the cognition of expert about "tall man". But this approach becomes difficult in many applications due to the following problems:

- Often a real numbers from the interval $[0,1]$ are not natural for a human expert, and it is very difficult for most experts to express their preference in a precise real number.
- The number of questions provided to experts may be infinite. For example it is not possible to ask an expert to assign a membership of "tall man" for everyone.

The most frequently used method in practice is to form a membership function from finitely many "real" memberships elicited from experts. For this purpose, only a small set of representative questions is delivered to experts, from which we reconstruct the desired function using interpolation, for example the simplest case of interpolation: *Piecewise linear functions*. For example, we want to build a fuzzy set for "tall man", since the height of a human being ranges between less than 1 meter to more than 2 meters, we pick up a set of 10 typical heights and may get exemplary memberships for "tall man" listed in Table 2.2. Based on these data, we need to use linear interpolation to find a "real" membership function. In general, the desired linear function, say A should pass through all measured point pairs (x_i, y_i), $\forall i \in \{1,\dots,10\}$, where x_i and y_i are the i-th height sample and membership provided by the expert respectively, that is,

$$A(x) = \begin{cases} 1, & \text{if } x > x_{10} \\ 0, & \text{if } x \le x_1 \\ y_{i-1} + \frac{x - x_{i-1}}{x_i - x_{i-1}} \cdot (y_i - y_{i-1}), & \text{if } x_{i-1} < x \le x_i,\ \forall i \in \{2,\dots,10\} \end{cases}$$

This function is plotted in Figure 2.3.

Table 2.2: Sampling of "tall man" for fuzzy set elicitation

Nr.	1	2	3	4	5	6	7	8	9	10
Height(m)	1	1.4	1.6	1.7	1.75	1.8	1.85	1.90	1.95	2
Membership	0	0	0.1	0.2	0.34	0.6	0.85	0.95	1	1

Obviously the more sampling provided, the more precise a piecewise linear function reflects the "real" membership function. On the other hand, it means more space needed to save the function, which is from user's and computer's point of view not favorable. More conveniently, a piecewise linear function can be used to approximate a membership function, which is determined by a few parameters. Regarding this, two other piecewise linear functions are usually applied to represent a membership function, namely *triangle* and *trapezoid* membership functions, as Figure 2.4 shows.

Definition 2.2 (Triangle membership function)**.** Given a numeric domain X and a fuzzy set A, a *triangle membership function* of A is characterized with three parameters ($l, m, h \in X$, and it holds $l \le m \le h$), and defined as:

$$A(x) = \begin{cases} \frac{h-x}{h-m}, & \text{if } m \le x \le h \\ \frac{x-l}{m-l}, & \text{if } l \le x < m \\ 0, & \text{otherwise} \end{cases}$$

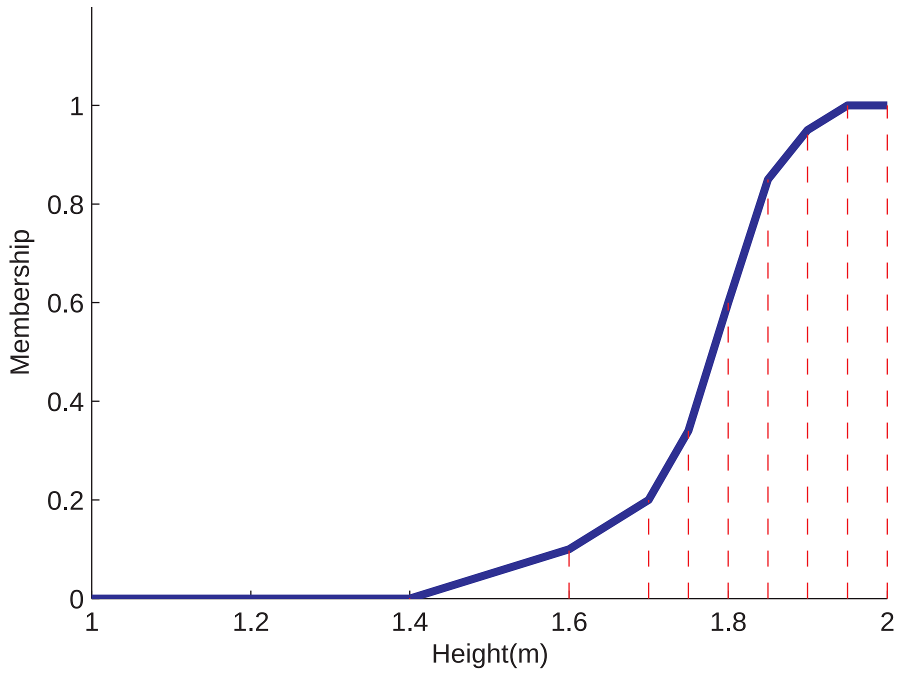


Figure 2.3: Piecewise linear membership function for "tall man"

shortly written as $A(x;l,m,h)$.

Definition 2.3 (Trapezoid membership function)**.** Given a numeric domain X and a fuzzy set A, a *trapezoid membership function* of A is characterized with four parameters ($l,m,n,h \in X$, and it holds $l \leq m \leq n \leq h$), and defined as:

$$A(x) = \begin{cases} \frac{h-x}{h-n}, & \text{if } n < x \leq h \\ 1, & \text{if } m \leq x \leq n \\ \frac{x-l}{m-l}, & \text{if } l \leq x < m \\ 0, & \text{otherwise} \end{cases}$$

shortly written as $A(x;l,m,n,h)$.

Furthermore we say a membership function is characterized by its *core* and *support*, where

- the *core* of a fuzzy set A on domain X is the crisp set that contains all elements of X that have membership degrees of one in A, that is:

  $$core(A) = \{x \in X : A(x) = 1\}$$

  In case of trapezoid membership function $A(x;l,m,n,h)$, the core is obviously an interval $[m,n]$;

- the *support* of a fuzzy set A on domain X is the crisp set that contains all elements of X that have nonzero membership degrees in A, that is:

  $$support(A) = \{x \in X : A(x) > 0\}$$

  In case of trapezoid membership function $A(x;l,m,n,h)$, the core is obviously an interval (l,h).

It is easy to see that trapezoid membership functions generalize triangle membership functions, in which the core of trapezoid membership functions shrinks to a crisp set with single element.

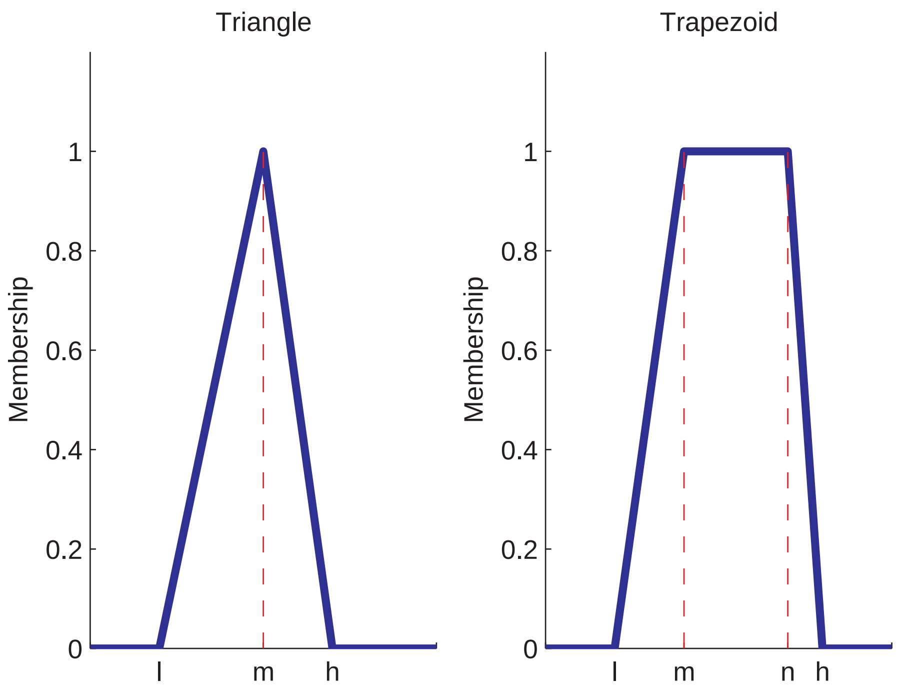


Figure 2.4: Triangle and trapezoid membership functions

Triangle and trapezoid functions are the simplest membership functions, since there are only three and four parameters needed to identify their functions. Nevertheless, they are also, at present, the most frequently used membership functions [NK02]. So far many current fuzzy systems and applications use them as the standard form of membership function, it seems that we do not need more complicated membership functions. Moreover, since we are approximating expert knowledge, it is not reasonable to try to formalize the fuzzy properties in a too complicated way. On the other hand, we should note that they are natural extensions of crisp tolerance intervals in traditional quality assessment by softening the lower and upper bounds of tolerance intervals. From this point of view, both are used as standard membership functions in this thesis.

The piecewise linear membership functions are only suitable for handling measurements under continuous and numeric domains, but not appropriate for discrete measurements, which are also very often used in practice. For example, a fuzzy set of "favored color" involves, after all, only discrete values (colors), as well as "preferred auto brand", "kind of sport" etc. The discrete membership functions are of course again discrete, and the memberships for all involved discrete values have to be elicited from experts particularly.

Definition 2.4 (Discrete membership function)**.** Given a discrete domain X with n distinct values (say $x_1, x_2, \ldots, x_n$) and a fuzzy set A, a *discrete* membership function A is characterized with n parameters $p_i \in [0,1], i \in \{1,\ldots,n\}$, so that

$$\forall i \in \{1,\ldots,n\} : A(x_i) = p_i$$

For example, a discrete membership function of fuzzy set "favored color" might have a form as illustrated in Figure 2.5.

As a special case of discrete membership function, an *ordinal* membership function is defined over an ordinal domain, in which values have a natural order. For example, in order to assess the satisfaction degree concerning the number of doors of autos, one might define a fuzzy set "big enough" on an ordinal domain with three values

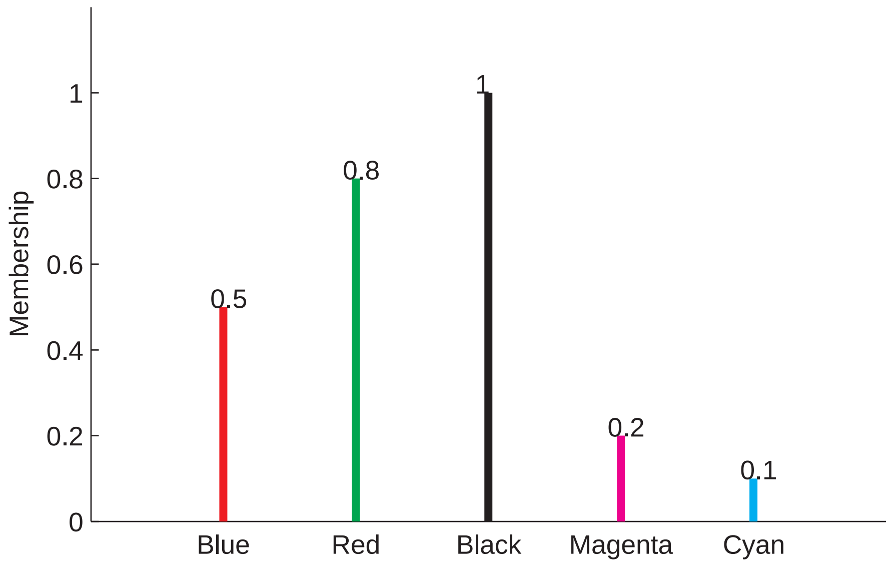


Figure 2.5: Discrete membership function for "favored color"

(2,3,4). A car with 4 doors is absolute "big enough", and has a membership 1. Furthermore, one might argue that 3 door-auto is preferred than 2 doors, but still worse than 4 doors. Figure 2.6 illustrates an ordinal membership function for fuzzy set "big enough" on ordinal number of doors.

In order to determine a fuzzy set on more than one variables, different types of membership functions are needed of course depending on concrete interest of the users, for example a two-dimensional matrix might be useful to define a fuzzy set with dimensionality two, etc.

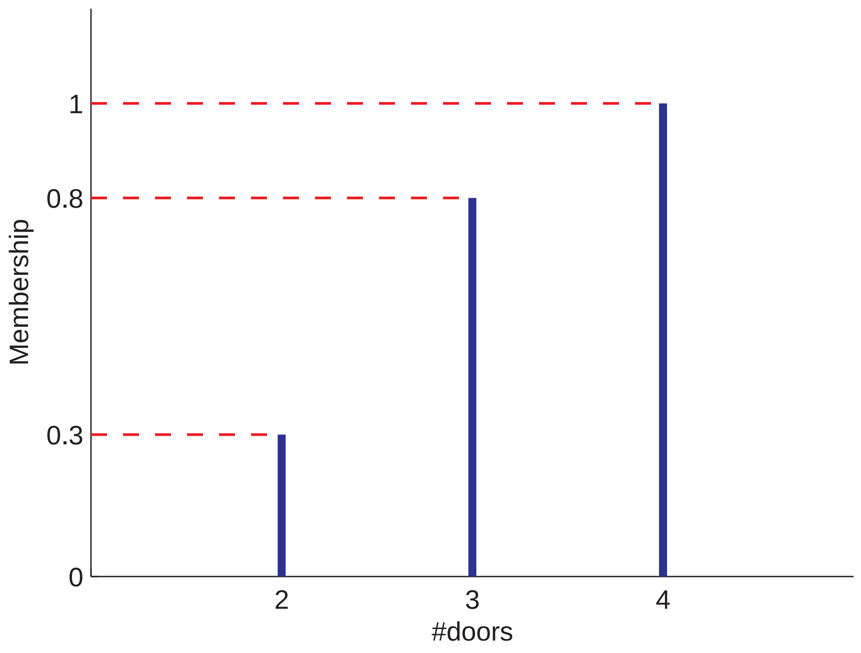


Figure 2.6: Ordinal membership function for "big enough" in terms of number of doors

2.2.2 Fuzzy Aggregation Operators

Fuzzy set theory also extends classical set-theoretic operations (intersection, union, etc.) for combining fuzzy sets [KY95]. Amongst others, we concentrate on three type of fuzzy aggregation operators: fuzzy *AND* , fuzzy *OR* and fuzzy *Average* (or fuzzy *Mean*).

Before we introduce the definitions of fuzzy *AND* and fuzzy *OR*, let us define two general functions to handle fuzzy membership degrees.

Definition 2.5 (T-norm)**.** A function $T : [0,1]^2 \to [0,1]$ is called a *triangular norm* (*t-norm*) if and only if it satisfies following conditions, for all $x,y,z \in [0,1]$:

- **Identity** : $T(x,1) = x$
- **Commutativity** : $T(x,y) = T(y,x)$
- **Associativity** : $T(x,T(y,z)) = T(T(x,y),z)$
- **Monotonicity** : $x \leq y \Rightarrow T(x,z) \leq T(y,z)$

Definition 2.6 (T-conorm)**.** A function $S : [0,1]^2 \to [0,1]$ is called a *triangular conorm* (*t-conorm*) if and only if it satisfies following conditions, for all $x,y,z \in [0,1]$:

- **Identity** : $S(x,0) = x$
- **Commutativity** : $S(x,y) = S(y,x)$
- **Associativity** : $S(x,S(y,z)) = S(S(x,y),z)$
- **Monotonicity** : $x \leq y \Rightarrow S(x,z) \leq S(y,z)$

With the help of the previous two functions, one can define the set-theoretic operators on fuzzy sets. Let A,B fuzzy sets on X, the fuzzy *AND* (denoted by $\wedge$) and *OR* (denoted by $\vee$) operators can be defined pointwise by using t-norm function $T : [0,1]^2 \to [0,1]$ and t-conorm function $S : [0,1]^2 \to [0,1]$ respectively, as follows:

$$(A \wedge B)(x) := T(A(x),B(x))$$
$$(A \vee B)(x) := S(A(x),B(x))$$

for all $x \in X$. It is simple to see that the Boolean algebra structure of classical sets does not hold under fuzzy sets [DP80].

For a t-norm T and a t-conorm S, the following properties can be implied from the definitions of the t-norm and the t-conorm:

- **Neutral element** : Since for all $x \in [0,1]$, it holds $T(x,1) = x$ and $S(x,0) = x$, 1 (0) is called the *neutral element* for T (S).
- **Absorbing element** : Since for all $x \in [0,1]$, it holds $T(x,0) = 0$ and $S(x,1) = 1$, 0 (1) is called the *absorbing element* for T (S).
- **Minimum and Maximum boundary** : For all $x \in [0,1]$, it holds $T(x,y) \leq \min(x,y)$ and $S(x,y) \geq \max(x,y)$

Definition 2.7 (Negation)**.** A function $N : [0,1] \to [0,1]$ is called a *negation*, if it satisfies following conditions:

- $N(0) = 1$;
- $N(1) = 0$;
- $\forall x,y \in [0,1] : x \leq y \Rightarrow N(x) \geq N(y)$.

Definition 2.8 (Strong negation)**.** A negation N is said to be *strong*, if it satisfied:

- N is continuous on x;
- N is strictly decreasing, namely $\forall x,y \in [0,1] : x < y \Rightarrow N(x) > N(y)$;
- N is involutive, namely $\forall x \in [0,1] : N(N(x)) = x$.

A common choice for a strong negation is defined by:

$$\forall x \in [0,1] : N(x) = 1 - x$$

which is also called *standard negation* and used in this thesis as default.

Without special remark, the t-norms and t-conorms are defined on two-dimensional space, the dimensionality of input can of course be extended thanks the associativity of both norms. Generally, a norm with high dimensionality n ($n > 2$) can be expressed in a recursive way as:

$$T(x_1, x_2, \ldots, x_n) = T(T(x_1, x_2, \ldots, x_{n-1}), x_n)$$

for $x_1, x_2, \ldots, x_n \in [0,1]$.

Definition 2.9 (De Morgan triplet)**.** A triplet (T, S, N) is called a *De Morgan triplet*, if T is a t-norm, S a t-conorm, N a strong negation and they satisfy the De Morgan's law:

$$\forall x,y \in [0,1] : S(x,y) = N(T(N(x), N(y)))$$

In the following text, we introduce several well-known t-norms and t-conorms in terms of the De Morgan triplet, as well as a parameterized norm family, in which the standard negation is taken as default, see Appendix A for a complete list of t-norm and t-conorm.

- **The minimum and the maximum**
  Zadeh proposed the use of min and max to define t-norm and t-conorm in his seminal paper [Zad65], which build a De Morgen triplet together with the standard negation, that is, for $x,y \in [0,1]$:

  $$\begin{aligned} T_{min}(x,y) &:= \min(x,y) \\ S_{max}(x,y) &:= \max(x,y) \end{aligned}$$

  As mentioned before, the T_{min} and S_{max} are the greatest t-norm and the smallest t-conorm. They are the most important norms both from a theoretical and from a practical point of view.

- **The product and the probabilistic sum**
  Goguen gives another De Morgan triplet in [Gog65], for $x,y \in [0,1]$:

  $$\begin{aligned} T_{pro}(x,y) &:= x \cdot y \\ S_{pro}(x,y) &:= x + y - x \cdot y \end{aligned}$$

Figure 2.7 to 2.8 illustrate the previously described t-norms and t-conorms in two-dimensional space.

Definition 2.10 (c-Idempotency)**.** A function $f(x_1, x_2, \ldots, x_n)$ is *c-idempotent*, if

$$\forall c \in [0,1] : x_1 = x_2 = \ldots = x_n = c \Rightarrow f(x_1, x_2, \ldots, x_n) = c$$

Figure 2.7: T-norms

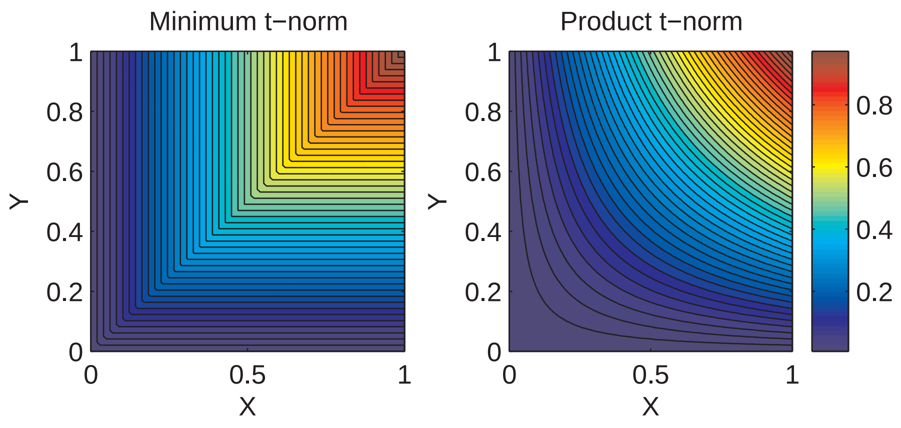
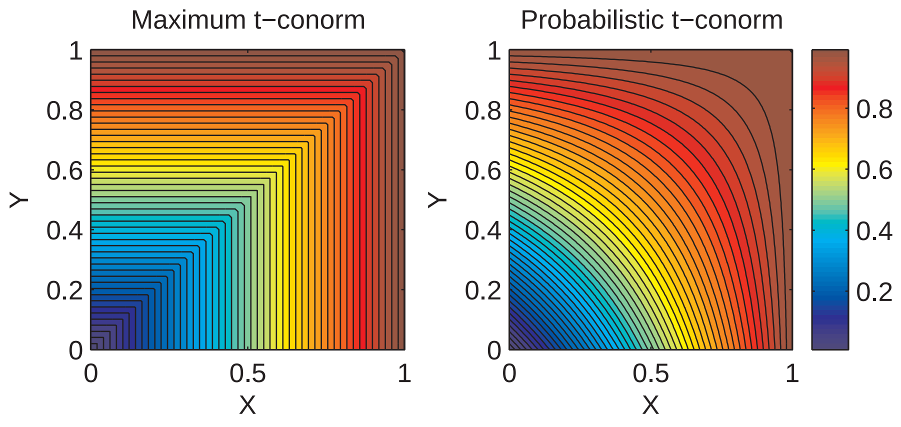


Figure 2.8: T-conorms



The idempotency of a t-norm and a t-conorm is not guaranteed from the definitions, but following propositions can be easily drawn without detailed proofs.

Proposition 2.11. *The minimum t-norm and the maximum t-conorm are c-idempotent for all $c \in [0,1]$, and they are the only ones with this property.*

Proposition 2.12. *All t-norms and t-conorms are 0-idempotent and 1-idempotent.*

Proposition 2.13. *All t-norms show the anti-monotonicity in dimensionality, namely:*

$$\forall x_1, x_2, \dots, x_n, x_{n+1} \in [0,1], \forall n \geq 1 : T(x_1, x_2, \dots, x_n) \geq T(x_1, x_2, \dots, x_n, x_{n+1})$$

where T is a t-norm.

Proposition 2.14. *All t-conorm show the monotonicity in dimensionality, namely:*

$$\forall x_1, x_2, \dots, x_n, x_{n+1} \in [0,1], \forall n \geq 1 : S(x_1, x_2, \dots, x_n) \leq S(x_1, x_2, \dots, x_n, x_{n+1})$$

where S is a t-conorm.

Adding arguments can never increase the output of a t-norm, whereas the output of a t-conorm can not be decreased by adding arguments to it.

The parameterized norms have gained more and more importance in fuzzy applications because of their flexibility. A well-known parameterized family of t-norm and t-conorm is, for example, the *Hamacher family* [Ham78]:

$$\begin{aligned} T_r(x,y) &:= \frac{x \cdot y}{r + (1-r)(x+y-xy)} \\ S_r(x,y) &:= \frac{x+y-(2-r) \cdot x \cdot y}{1-(1-r) \cdot x \cdot y} \end{aligned}$$

for $x, y \in [0,1]$ and a parameter $r \geq 0$. The Hamacher family is employed in this thesis because of its simplicity and other nice properties (see the identifiability in Section 3.2) as standard t-norm and t-conorm. Even though, all families listed in Appendix A can be applied in principle. As the parameter r changes, we can obtain different norms from the Hamacher family, such as:

$$r = \begin{cases} 1 & T_r \text{ is the product t-norm} \\ 2 & T_r \text{ is the Einstein t-norm} \\ \infty & T_r \text{ is the drastic t-norm} \end{cases}$$

From the definition of the Hamacher family, one can follow the continuity of Hamacher t-norms (t-conorms), that is:

Proposition 2.15. *A Hamacher t-norm (t-conorm) $T_r(x,y)$ ($S_r(x,y)$) is continuous on* $[0,1] \times [0,1]$ *with respect of x and y.*

Since t-norms and t-conorms are, respectively, conjunctive and disjunctive aggregating, which do not make any compensation. For example, given two membership degrees $x = 0.3$ and $y = 0.5$, the overall membership $T(x,y)$ ($S(x,y)$) is restricted to the interval $[0, 0.3]$ ($[0.5, 1]$) for any t-norm T (t-conorm S). Often a type of fuzzy operator lying somewhere between the t-norm and the t-conorm is desired. Here, we introduce another type of fuzzy aggregation operator, called the *fuzzy average*.

Definition 2.16 (Mixnorm)**.** a function $M : [0,1]^2 \to [0,1]$ is called a *fuzzy average* (*mixnorm*), if it satisfies following conditions:

- **Commutativity** : $\forall x, y \in [0,1] : M(x,y) = M(y,x)$
  the value of a mixnorm is indifferent to the ordering of its arguments.
- **Monotonicity** : $\forall i \in \{1, \ldots, n\} : x_i \geq x_i' \Rightarrow M(x_1, \ldots, x_i, \ldots, x_n) \geq M(x_1, \ldots, x_i', \ldots, x_n)$.
- **Idempotency**
  a mixnorm is c-idempotent for all $c \in [0,1]$.

From the definition of mixnorm, one can directly imply that a mixnorm always lies between the min and max of the arguments, it holds for any mixnorm M:

$$\min(x,y) \leq M(x,y) \leq \max(x,y)$$

Obviously, the arithmetic mean $\hat{x} := \frac{1}{n} \sum_{i=1}^{n} x_i$, as well as min, max and median are mixnorms. Yager proposed a mixnorm called the *ordered weighted averaging* (OWA) in [Yag88].

Definition 2.17 (OWA mixnorm)**.** An OWA mixnorm of dimension n is a mapping:

$$M_{OWA} : [0,1]^n \to [0,1].$$

A n-vector W is associated with M_{OWA}:

$$W := \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_n \end{bmatrix}$$

so that:

1. $w_i \in [0,1]$, for all $i \in \{1,\ldots,n\}$,
2. $\sum_{i=1}^{n} w_i = 1$.

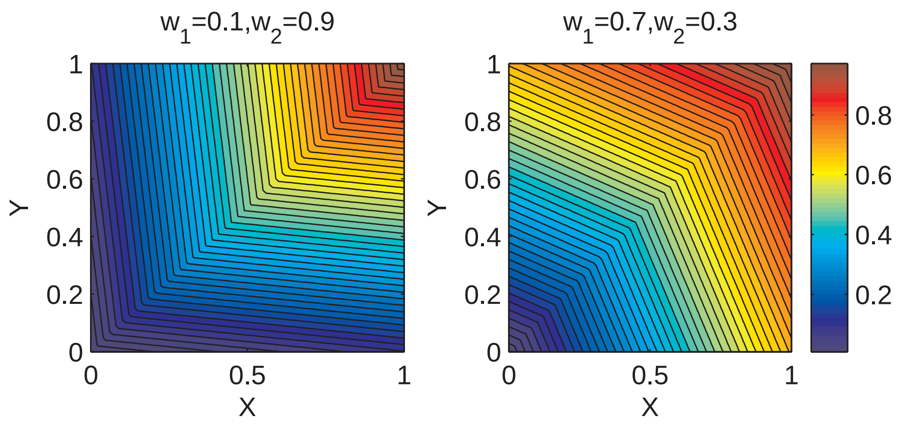
Then we have:

$$M_{OWA}(x_1,x_2,\ldots,x_n) = \sum_{i=1}^{n} w_i \cdot y_i$$

where y_i is the i-th largest element in $\{x_1,x_2,\ldots,x_n\}$.

An OWA mixnorm is parameterized with a weighting vector, which is not associated with a finite component x_i, but with a particular ordered position of x_i. A fundamental aspect of the OWA operator is the re-ordering step, this introduces nonlinearity into the aggregation process. An OWA mixnorm allow us to aggregate fuzzy membership degrees in an easily, convenient way. Figure 2.9 illustrates the OWA mixnorms in two-dimensional space with given weighting vector.

Figure 2.9: OWA mixnorms



It is noted that different OWA mixnorms are distinguished by their weighting vector, specially with a weighting vector $W = [1,0,\ldots,0]'$ an OWA mixnorm turns to the max function, and with a weighting vector $W = [0,\ldots,0,1]'$ to the min function. Furthermore, an OWA mixnorm can serve as the *mean* or *median* operator with a proper weighting vector [FY02], etc. The OWA mixnorms fill the gap between conjunctive and disjunctive combinations, and provide flexible compensations among fuzzy membership degrees.

From the definition of OWA mixnorms, one can follow the continuity of OWA mixnorms.

Proposition 2.18. *An OWA mixnorm $M_{OWA}(x_1,\ldots,x_n;w_1,\ldots,w_n)$ is continuous on each variable x_i.*

Putting all three aggregation operators together, the boundary property among them can be easily seen, namely:

$$\forall x,y \in [0,1] : T(x,y) \leq \min(x,y) \leq M(x,y) \leq \max(x,y) \leq S(x,y)$$

where T stands for t-norm, S for t-conorm and M for mixnorm (see Figure 2.10). We thus obtain a continuous spectrum of aggregation operators, in which we can express any kind of combination between membership degrees.

2.2.3 Linguistic Hedges

In order to weigh sub-criteria, we apply the linguistic hedges in this thesis, which is a function $H : [0,1] \to [0,1]$ that depicts the effect of linguistic hedges upon a

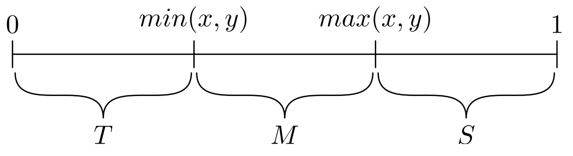


Figure 2.10: Boundary properties of t-norm, t-conorm and mixnorm

fuzzy membership degree [Zad72, Lak73]. Note that the linguistic hedges have a nice property that they can be well interpreted in the natural language, a linguistic hedge is a qualifier on a linguistic proposition, which is very widely used in natural language of human being, like: “a very very old man”, “he is slightly drunk” or “x is approximately 4” etc. Among several approaches proposed for the representation of linguistic hedges (for instance Zadeh in [Zad73], De Cock in [Coc02] and Nguyen in [NH02]), we shall consider a suggestion from Zadeh [Zad73], where linguistic hedges can be classified into *concentration* and *dilation*, depending on in which way it modifies membership of a fuzzy object.

- **Concentration**
  A linguistic hedge H of type concentration on a fuzzy set A results in a reduction in the magnitude of the memberships of x in A, which is relatively small for those x with a high membership in A and relative large for those x with low membership. It can be defined as:

  $$H(A(x)) := (A(x))^{\alpha}$$

  where $\alpha > 1$. Note that the bigger the parameter α, the more a membership function concentrates on its core, as Figure 2.11 shows. A linguistic hedge of type concentration can be interpreted in natural language with terms such as *absolutely*, *very*, *much more*, *more* and *a little more* under specifying the values of α as $4, 2, 1.75, 1.5$ and 1.25 respectively [CL03].

- **Dilation**
  Reversely, a linguistic hedge H of type dilation on a fuzzy set A increases the magnitude of the memberships of x in A, which is relatively small for those x with a high membership in A and relative large for those x with low membership. It can be defined as:

  $$H(A(x)) := (A(x))^{\alpha}$$

  where $0 < \alpha < 1$. Note that the smaller the parameter α, the more a membership function dilates its support , as Figure 2.12 shows. A linguistic hedge of type dilation can be interpreted in natural language such as *a little less*, *more or less* and *slightly* under specifying the values of α as $0.75, 0.5, 0.25$ respectively [CL03].

From the technical point of view, linguistic hedges involved in an FOT are all initialized with parameters 1, that is, a linguistic hedge servers as an identify function by default, in which case we say a linguistic hedge is switched off. Of course, the parameter can be tuned later by human experts or other automatic calibration process.

From the definition of linguistic hedges, one can easily follow the idempotency of the linguistic hedges, namely:

Proposition 2.19. *All linguistic hedges show the* 1- *and* 0-*idempotency.*

As an example, we consider a fuzzy set “comfortable temperature” A under the assumption that the most “comfortable temperature” lies at 25°, under 20° or above

30° are not "comfortable" at all. Based on these assumption, we build a triangle membership function for A. Figure 2.11 illustrates the effect of linguistic hedges of type concentrations, and Figure 2.12 illustrates the effect of linguistic hedges of type dilation.

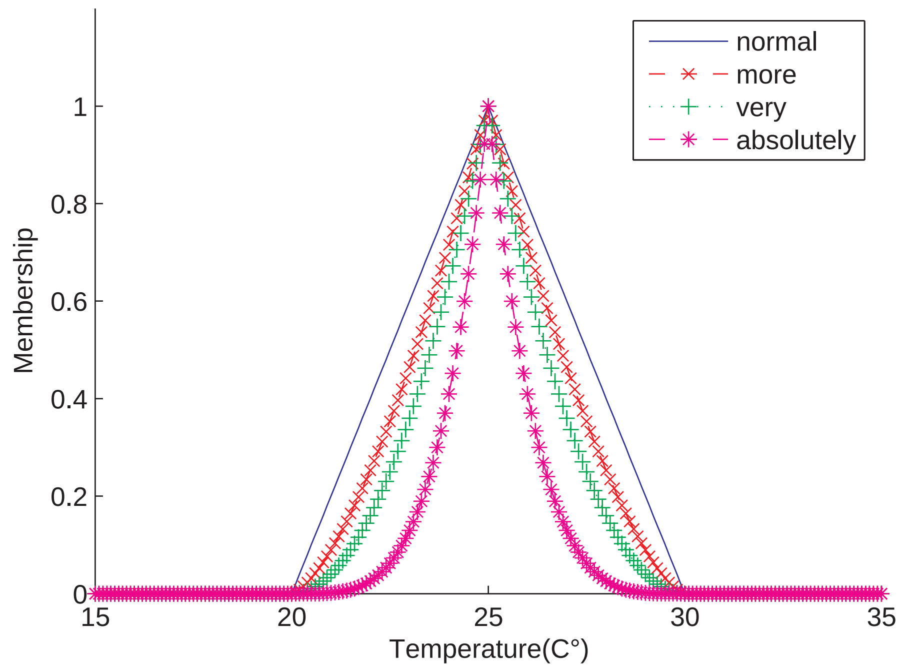


Figure 2.11: Linguistic hedges of type concentration for "comfortable temperature"

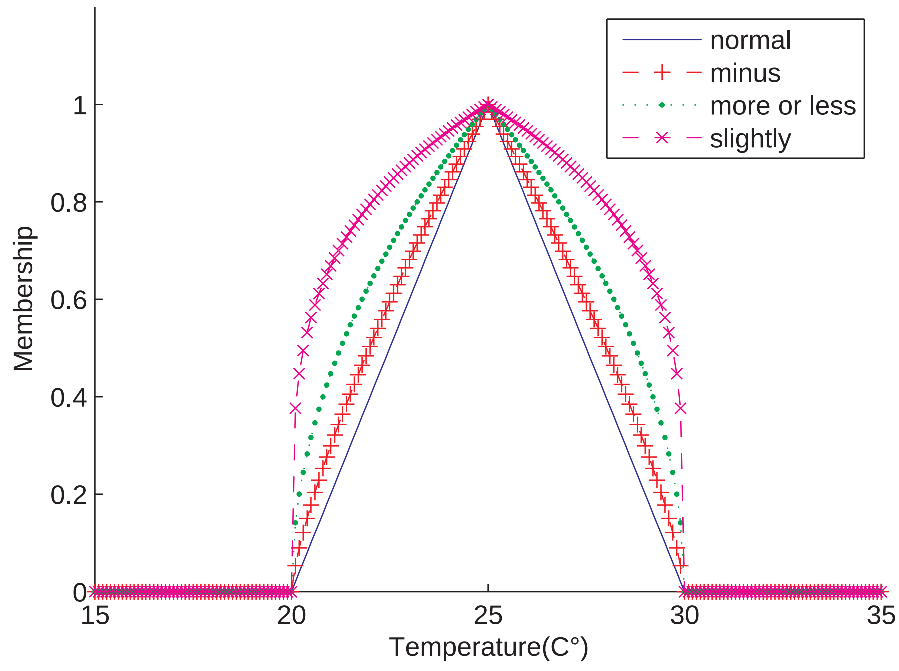


Figure 2.12: Linguistic hedges of type dilation for "comfortable temperature"

2.3 Fuzzy Operator Tree and Properties

Having introduced the main components from fuzzy set theory, we can consider the fuzzy operator tree in this section. The properties of an FOT are described in the subsection 2.3.1, then we give a more detailed example of an FOT, at the end of this section the issue of handling discrete utility degrees with an FOT is addressed.

2.3.1 Formal Definition and Properties of Fuzzy Operator Tree

In this thesis, we restrict us to only use the Hamacher t-norm and t-conorm, as well as the OWA mixnorm, as reasoned later in Section 3.2. For the sake of clearness, we take an abstract term *operator* to refer a Hamacher t-norm (T_h), a Hamacher t-conorm (S_h) or an OWA mixnorm (M_{OWA}) in following text.

Definition 2.20 (Fuzzy operator tree)**.** A *fuzzy operator tree* (FOT for short), say F, is a mapping from the domain of alternatives to be evaluated O to the unit interval $[0,1]$, namely:

$$F : O \rightarrow [0,1]$$

which can be defined in the Backus-Naur notation by:

$$\begin{aligned} <F>::= &\ <A>(<H>\{,<H>\}) \mid f \\ <H>::= &\ h(<F>) \mid\ <F> \\ <A>::= &\ T \mid S \mid M \end{aligned}$$

in which:

- T,S,M stand for a parametrized Fuzzy Operator;
- f is a Fuzzy Set;
- h is a linguistic hedge.

An FOT models a utility function and assigns a utility degree in $[0,1]$ for an input object o, which is represented by a feature vector ($o = [x_1, x_2, \dots, x_n]$). An FOT can be represented in a tree-like structure, which is characterized with following properties:

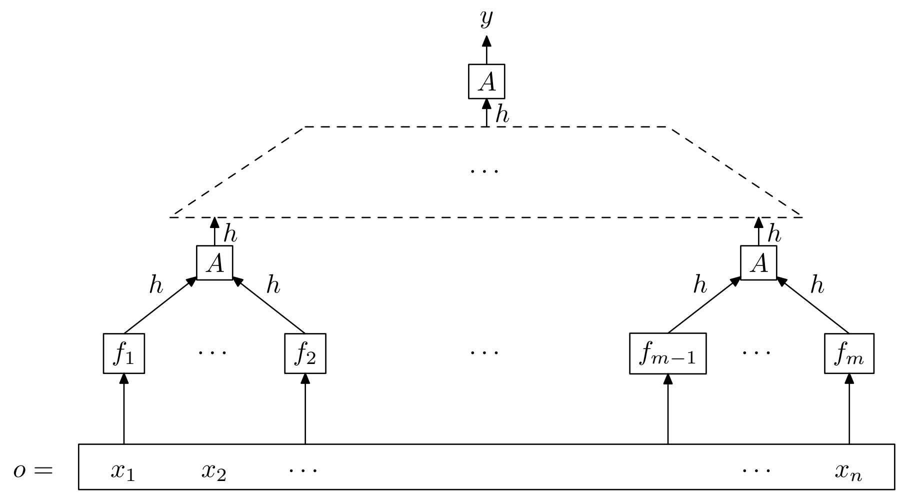
- Its leaf node represents a basic evaluation, which is associated with a fuzzy set;
- Its interior node, including the root node, corresponds to an operator and its parameter(s) respectively; an interior node takes the outputs of its child nodes as inputs, and delivers an aggregated evaluation to the parent node;
- An edge connects a child node to its parent node and is associated with a linguistic hedge (marked with the respective parameter);
- The root of FOT gives the overall utility degree.

Figure 2.13 illustrates an FOT schematically, in which the y denotes the overall utility degree of the input vector o ($o = [x_1, x_2, \dots, x_n]$), $f_1, f_2, \dots, f_m$ are m fuzzy sets over o, A stands for an operator and h indicates a linguistic hedge.

Note that objects to be assessed by FOTs are assumed to be represented in the form of a feature vector, which is the simplest, common-used form to characterize objects. There are several other alternatives to represent objects, for instance, time series are used to describe measurements taken at consecutive times, graphic or image data are located in multi-dimensional space, etc. Usually, it is crucial to be aware of the representation of the data, any proper form of representation can be employed here, but for convenience of analyze, we assume that the measurement data for FOTs are represented in the form of feature vector.

Fuzzy sets in an FOT serve to modulate basic evaluations on the feature vector, such that an FOT can serve to aggregate evaluations in a hierarchical, flexible way in order to give one overall evaluation finally. The type and definition of the fuzzy set depends

Figure 2.13: Schematic illustration of an FOT



strongly on the special interest of the human expert in concrete applications. As standard case, we assume that the fuzzy sets involved in an FOT are defined by a triangle or trapezoid membership function based on a single numeric input variable. But in principle, fuzzy sets on discrete or ordinal variable are of course allowed, as well as fuzzy sets on multiple variables. Furthermore, fuzzy sets in an FOT can share input variables, which is also often observed in practice. Generally speaking, any kind of fuzzy set can be applied to construct an FOT based on any form of input data, as long as they can express basic evaluations for the overall quality assessment.

Related to the FOT, following terms are used in this thesis:

- **Height**
  The height of an FOT is defined as the number of passed nodes on longest downward path from root node to a leaf node (fuzzy set), including the root node.

- **Complexity**
  There are two kind of complexities related to an FOT: The *component complexity* gives the number of components included in an FOT, that is, the number of operators, linguistic hedges and fuzzy sets, and the *parameter complexity* is defined as the number of all parameters in an FOT, since one component of an FOT might have more than one parameter, the parameter complexity of an FOT is usually bigger than its component complexity. As default, the complexity of an FOT denotes its component complexity in this thesis.

Taking a closer look at the definition of the FOT indicates several important properties associated with FOT:

1. an FOT shows its *continuity* on each basic evaluation in the unit interval $[0,1]$, if all basic evaluations based on the corresponding fuzzy sets are continuous in the unit interval $[0,1]$. This can simply be drawn from the continuity of fuzzy aggregation operators and linguistic hedges;

2. an FOT shows its *monotonicity* on the Cartesian product of $[0,1]$ domains of all basic evaluations based on fuzzy sets, which can be easily proved due to the monotonicities of fuzzy aggregation operators, linguistic hedges;

3. the number of fuzzy sets m, operators p and linguistic hedges q holds:

$$q = m + p - 1$$

that is the number of linguistic hedges q is the sum of number of fuzzy sets m and number of norms p minus 1 in an FOT, which can be implied from the following two facts:

(a) each fuzzy set is modified by a linguistic hedge, which is illustrated by an edge between the fuzzy set and its parent operator;

(b) each fuzzy aggregation operator is modified by a linguistic hedge in turn, except the root node.

We would like to emphasize again one of the attractive advantages of using an FOT, namely its hierarchical modular structure, which allows one to specify utility functions in a recursive and systematic way. By the quality assessment of technical products, the main motivation of this thesis, their modular structure can be perfectly reflected in an FOT. Typically, a technical product itself has a modular structure and can be decomposed into subcomponents in a recursive manner. For example, the quality of a car depends on, say, the engine system, the electronic control system, etc. Amongst others, the electronic control system consists again of a number of components (GPS, car radio, etc.), which in turn is made up of several subcomponents, and so on. We can employ an FOT to model the quality function for a car, at the same time also for its subcomponents. From this point of view, a subcomponents represented by an FOT is just a sub-tree in a super ordinate FOT, in other words, an FOT can be plugged into another FOT by replacing one of its basic evaluations. As we have seen from this example, the "divide-and-conquer" strategy can be successfully carried out by using FOTs, which makes the assessment of very complex systems controllable.

2.3.2 An Example of Fuzzy Operator Tree

Again, we consider the example to assess candidates for a certain job, which is already mentioned in Section 1.2.1 and shown in Figure 2.14. Here we use $\wedge$ to denote a Hamacher t-norm, $\vee$ for a Hamacher t-conorm and $\varnothing$ for an OWA mixnorm, in addition there are four fuzzy sets $(f_1(x_1), \ldots, f_4(x_4))$ for the basic evaluations on math, CS, French and English of the candidate, in which the corresponding grades are taken to build input vector$([x_1, \ldots, x_4])$. Finally there are six linguistic hedges $(h_1, \ldots, h_6)$ involved in this case.

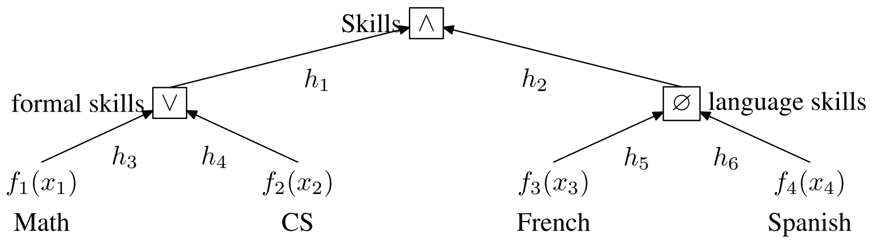


Figure 2.14: FOT example for the assessment of a candidate

We assume that all operators and linguistic hedges are determined with following parameters:

- $\wedge$ is the Hamacher t-norm with parameter 0.4;
- $\vee$ is the Hamacher t-conorm with parameter 1.2;
- $\varnothing$ is the OWA mixnorm with one parameter 0.7;
- $h_1, \ldots, h_6$ are assigned with $0.5, 4, 2, 2, 1.5, 2$, which can be interpreted into natural language terms *more or less*, *absolutely*, *very* and so on.

This FOT can be expressed formally:

$$F(x_1,x_2,x_3,x_4) = T_h(h_1(S_h(h_3(f_1(x_1)),h_4(f_2(x_2)))),h_2(M_{OWA}(h_5(f_3(x_3)),h_6(f_4(x_4)))))$$

We like to highlight the good interpretability of FOTs here again. Although the formal expression of this example seems not very comprehensible at the first glance, let us consider the corresponding knowledge expressed in the form of natural language:

- A good candidate should have *more or less* good formal skills *AND absolute* good language skills, in more detail good formal skills mean *very good* mathematics *OR very good* computer knowledge, and good language skills are *averaged* between *very good* French and slightly good Spanish.

As we can see, the interpretation of an FOT in the form of natural language is very familiar to human being, and can thus be understood very well. No matter the increased complexity of an FOT, the good interpretability is kept in any case due to the modularity of an FOT.

2.3.3 Discrete Values in Fuzzy Operator Tree

Originally an FOT is designed to take numeric inputs and give numeric utilities in the unit interval $[0,1]$, but in practice, an FOT has to handle not only numeric variables, but also discrete ones. For example, the basic evaluations for fuzzy sets are sometimes discrete measurements, such as the number of doors of a car, on the other hand one might prefer to a discrete quality category rather than real-valued evaluation as output of an FOT, such categories like quality class A (top-quality), B (still OK) or C (not acceptable), etc.

The discrete and ordinal membership functions introduced before are designed to handle the discrete measurements. For a discrete output, without change on the interior structure of the FOT, we consider to convert a discrete input into a numeric one, and reversely a numeric output into a discrete one. The former transformation is already achieved in an FOT by using discrete or ordinal membership functions as mentioned before. For the later transformation, a *discretizer* would be employed here on the top of an FOT, which, as its name indicates, discretizes a real-value utility into predefined set of categories.

In this thesis, a simple “split” discretizer is applied to realize such a discretization, in which $k+1$ thresholds are defined $(t_0,t_1,\ldots,t_k)$ for k categories. It holds:

$$0 = t_0 < t_1 < \ldots < t_k = 1$$

Formally this kind of discretizer (noted as D) is a mapping

$$D : [0,1] \to \{1,2,\ldots,k\}$$

where k is the number of available categories. Together with the $k+1$ thresholds, the “split” discretizer is defined by:

$$D(y) = \begin{cases} j & \text{if } y \in (t_{j-1},t_j] \\ 1 & \text{if } y = 0 \end{cases}$$

2.4 Fuzzy Operator Tree Elicitation

This section is contributed to construct an FOT with the help of experts. To guarantee the interpretability of an FOT, an automatic method without using an expert's guide is not desirable here. So how to help experts to express their prior knowledge in the form of an FOT, or shortly FOT elicitation, becomes the first step for applications with FOTs. From the definition of the FOT, there are the following tasks for this section:

- **Fuzzy sets elicitation**
  The definitions of fuzzy sets and their membership functions have to be determined, which strongly depend on the applied area and particular interests of applications, rarely two applications share identical fuzzy sets. We describe the related methods in Section 2.4.1.

- **Interior structure elicitation**
  The interior structure of an FOT describes the interrelationship among fuzzy sets, or aggregations on fuzzy sets regarding to the overall utility, which are organized in a recursive hierarchical tree-like structure. The challenge here focuses on the determination of height and component complexities of an FOT, as well as the type of aggregation operators. Section 2.4.2 is devoted to this topic.

- **Linguistic hedges determination**
  In contrast to two former components, the number of linguistic hedges involved in an FOT is fixed, if the number of fuzzy sets and aggregation operators has been determined. Therefore we just need to assign "proper" parameters for the linguistic hedges in an FOT. This can be done using expert knowledge, if it is available. The elicitation process in this case is relatively straightforward comparing with the former two components. We refer to [CR06, Lak73] for more details. Otherwise the parameter specification can be realized by the calibration process discussed in the next chapter.

2.4.1 Fuzzy Set Elicitation

Here we intend to extract the number of necessary fuzzy sets to express the interest of human experts, then their membership functions would be determined. At the end of this subsection, we illustrate this elicitation procedure using an example.

2.4.1.1 Fuzzy Properties Elicitation

Since a fuzzy set expresses a property of an object in a form of "object X is ...", the following question is going to be answered in this section: Which properties of objects an expert is interesting during the assessment process? For example, if an expert summaries his experience so:

The quality of the candidate depends on mathematics and communication skills,

which can be interpreted as:

A candidate with good utility should have good mathematics and good communication skills.

From this example, we are going to construct two fuzzy sets "good mathematics" and "good communication skills", as long as one can directly read "mathematics" and "communication skills" measurements from input vector, such as school grade, or certification grade and so on. Otherwise if a term can not be extracted from the input vector directly, such as “communication skills” here, which is again aggregated by other basic measurements, one might split this term more detailed, and we shall elaborate this approach in Section 2.4.2.

Note that the triangle and the trapezoid fuzzy sets, which are suggested in this thesis to handle the continuous measurements, are unimodal membership functions, and not appropriate for those fuzzy multimodal properties, for example a fuzzy set for “unusual temperature”, for which one might argue that a temperature over 50° or beneath -20° are unusual. To handle a multimodal property in an FOT, we suggest to decompose it into several unimodal ones, then aggregate them using an aggregation operator.

In most cases a basic measurement in the form of an input feature vector corresponds to a fuzzy set directly, on which a basic evaluation is carried out. For instance in the “candidate” case described in Section 2.3.2, the four leaf nodes (“Math”, “CS”, “French” and “Spanish”) are four independent fuzzy sets, and they can be described by piecewise linear membership functions, as mentioned before. However there are some cases, where a basic evaluation on measurements is some kind of relationship on more than one measurements, such as in the car radio example described in Section 1.1.2, where one might be interest whether three buttons with similar functionalities are equally operable. In such cases, we have to extend the dimensionality of fuzzy sets to express fuzzy relations with multi-dimensional membership function, such as a membership function in three-dimensional space to describe whether three buttons are equally operable. This kind of multi-dimensional membership function can be defined in very various manners, therefore, in the following text and experiments, we restrict our works on the fuzzy sets with one-dimensional piecewise linear membership function.

2.4.1.2 Membership Function Determination

As long as experts have decided the set of fuzzy sets to be involved in an FOT, it is necessary to determine the membership functions for each of these fuzzy sets. These functions should be elicited from experts again. The empirical researches on membership function elicitation had been conducted in [KB74, KGK95, BT02, Saa74, ITS00].

Since we limit us to one-dimensional piecewise linear membership functions that are parameterized by several points, constructing membership functions aim to locate these points precisely as well as possible. Usually, the following methods are used in experiments to elicitate membership functions:

- **Direct rating**
  The most straightforward way to detect a membership function seems to directly ask an expert the memberships for selected points, for example, a typical question for direct rating might be:

  What are the memberships for the following examples? ...

  At the end we can simply construct a piecewise linear membership function by connecting all known points, or at least approximate them with a mathematical model such as trapezoid function. The difficulties using direct rating is that experts might not be able to distinguish tiny differences between several points with similar memberships. Beside this, the performance of direct rating suffers individual subjective vagueness, for instance, the decision of an expert can be influenced by past answers, in this case the order of points provided to the expert affects the form of the membership functions.

- **Reverse rating**
  In this method, an expert would be asked to identify the object for which a given membership corresponds, for example:

  Who, do you think, is 0.6 *tall?*

  However this method has the same difficulties as direct rating.

- **Interval estimation**
  Instead of points, interval estimation subscribes to the random set-view of the membership function [1] The subject is asked to give an interval that describes the "α-cuts". A pair of a set-valued observations (in the form of an interval) and the corresponding frequency it was observed defines a random set for determining the membership function. A typical question sounds like:

  In which interval does a fuzzy set have a membership degree in $[0.7, 1]$ *(a* α*-cut with* $\alpha = 0.7$*)?*

  Notice that this method is more appropriate to those situations where a clear linear order in the measurement of the fuzzy concept exists, for example tallness, heat or time, etc. Interval estimation is a relatively simple way of acquiring the membership function, which is "less fuzzy" (the spread is narrower) compared to direct rating and reverse rating. Interval estimation subscribes to the uncertainty view of membership functions as opposed to the vagueness view and in that sense it brings the issues of uncertainty modeling using fuzzy set theory, random sets, possibility measures and their relations to probability theory [DP93].

- **Pairwise comparison**
  Kochen and Badre [KB74] report experimental results for the "precision" of membership functions using pairwise comparison method. Later Chameau & Santamarina [Cha87] also use the pairwise comparison technique and report it to be as robust as direct rating. Typical questions for pairwise comparison are:

  Which color do you prefer, blue or green?

  , and after the answer to this question (say, blue is chose):

  How much do you prefer blue to green?

  Pairwise comparisons can yield a matrix of relative weights between selected representative points. The membership function can be found by taking the components of the eigenvector corresponding to the maximum Eigenvalue.

Actually, the previously described methods are integrated in many cases. Which method(s) is (are) used depends on the concrete problem. While one method is selected to determine a membership function for a fuzzy set, other method can be applied to validate the elicitation process. Furthermore we allow a fine tuning of membership functions in a data-driven way, in which a membership function is treated

[1] Specially, one can also view the membership function horizontally where a fuzzy set A on domain X is represented in terms of its "level-cuts": $\{A_\alpha : \alpha \in (0,1]\}$ with $A_\alpha = \{x \in X : A(x) \geq \alpha\}$, as Zadeh in [Zad71] defines the membership function as: $A(x) = \sup \alpha \in (0,1] : x \in A_\alpha$.

as parameterized, and its parameters are allowed to be tuned during the optimization process, which will be discussed later.

As we have mentioned before that the trapezoid membership function is the main type of membership functions used in this thesis for quality assessment, the simplest method to determine a trapezoid membership function is to ask for two intervals (*core* and *support*). On the other hand, regarding quality assessment, intervals are the commonly used form to express the constraints on criteria, so the standard method in this thesis is just to elicit two intervals from human experts in order to determine the trapezoid membership function of a fuzzy set.

2.4.2 Interior Structure Elicitation

The determination of interior structure of an FOT is extremely problem-dependent, so automatic or semi-automatic construction of an FOT requires a deep understanding of the faced problem. Only several researches offer quite a limited support for this stage, such as [BR90]. In contrast to an automatic approach, the interior structure of an FOT would be elicited from experts in related field. In the remainder we introduce two elicitation methods at first, and then give a more detailed example at the end of this section.

2.4.2.1 Bottom-Up Construction

As the name suggested, a bottom-up manner for interior structure elicitation begins with a candidate set consisting of all available basic criteria, which are associated with fuzzy sets extracted by the elicitation methods mentioned before. The expert is asked to aggregate several criteria contained in the candidate set according to their relationship, this step corresponds to create a fuzzy aggregation operator for the FOT, in which several criteria in lower level are aggregated into a higher level criterion. The newly created operator corresponds the fuzzy connective terms of “AND”, “OR” and “Average” in the natural language. At the same time, the parameter(s) involved in newly created operator has to be given by human expert. After this, the already used criteria are removed from candidate set, while the newly created operator is added into candidate set as a new criterion. The aggregation step is repeated until the overall quality can be aggregated using criteria from the candidate set. This idea is detailed in Algorithm 2.1.

Algorithm 2.1: Interior Structure Elicitation: A bottom-up manner

```
   Input: Basic criteria associated with fuzzy sets $M$, Expert $E$
   Output: FOT $F$
1  begin
2      initialize $F$ with $M$ ;
3      $c \leftarrow \{m | m \in M\}$ ;
4      repeat
5          inquiry on $E$ to aggregate several criteria $a$ from $c$ ;
6          create new aggregation operator $n$ based on $a$ ;
7          $c \leftarrow c \setminus a$ ;
8          $c \leftarrow c \cup n$ ;
9          insert $n$ into $F$ as parent node of $a$;
10     until (The overall quality can be aggregated using criteria from c) ;
11     create aggregation operator $n$ based on $c$ ;
12     insert $n$ into $F$ as root node;
13 end
```

Note that in the line 5, a new aggregation operator would be generated with the help of a human expert, the specification of the parameter(s) in the operator may be relatively straightforward in many cases, for instance, in the quality assessment of technical products, the parameterization is usually predefined and can simply be taken from a specification sheet. However, in other cases, the expert may have problems specifying precise parameters. We shall discuss this problem in the following chapter and propose a new method that free the expert from this. This method is illustrated in Figure 2.15, in order to elicit the interior structure based on the candidate example described in Section 1.2.1.

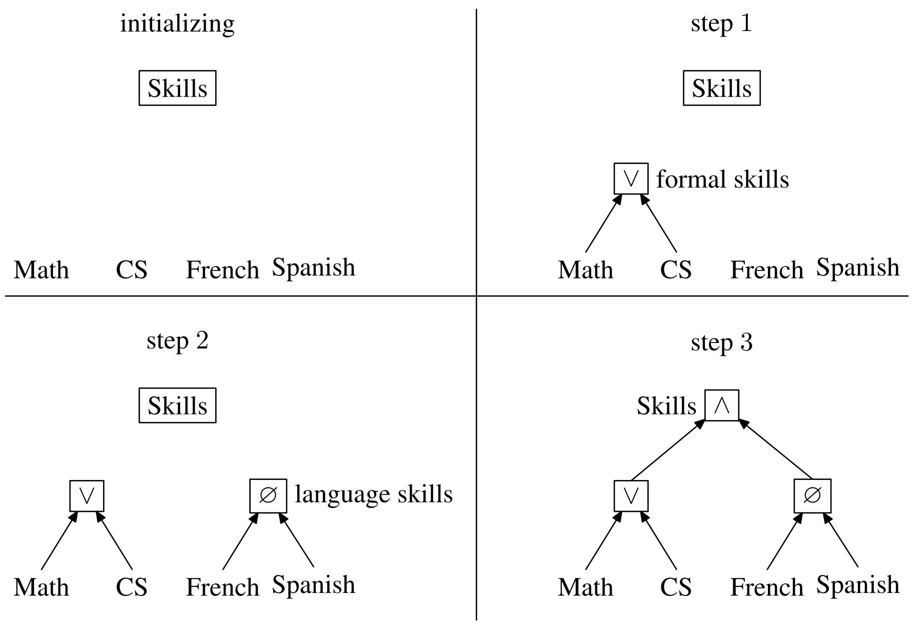


Figure 2.15: Interior structure elicitation in a bottom-up manner

2.4.2.2 Top-Down Construction

This method starts with the overall utility of alternatives being evaluated, in which an expert is inquired to express the utility as an aggregation of directly related criteria. As long as one of these criteria does not correspond to one basic criterion associated with the predefined fuzzy set, the expert is asked to decompose it further into several sub-criteria. This process is repeated until every criterion is either aggregated by other sub-criteria, or is a basic criterion already. This idea is detailed in Algorithm 2.2. This method is illustrated in Figure 2.16, in order to elicit the interior structure based on the candidate example described in Section 1.2.1.

2.4.2.3 An Example of Interior Structure Elicitation

As a concrete example, we consider a car selection problem introduced by Bohanec and Rajkoviă [BR88]. In order to assess the quality of cars there are six attributes available to describe the status of cars, let us assume each attribute is modulated with a fuzzy set, which has been defined using fuzzy sets elicitation methods mentioned before, Furthermore we assume that the fuzzy set is named by the corresponding attribute, a fuzzy set serves as a basic criterion on the related attribute. These six attributes, their descriptions and fuzzy sets are listed in Table 2.3.

In order to construct an FOT for the car selection problem in a bottom-up manner, Table 2.4 demonstrates the elicitation process in four steps, where the available candidate set is listed at first for each step, then several sub-criteria from the candidate set would be aggregated into a new criterion, the second column shows the aggregation

Algorithm 2.2: Interior Structure Elicitation: A top-down manner

```
Input: Fuzzy sets $M$, Expert $E$
Output: FOT $F$
1  begin
2      $F \leftarrow \emptyset$ ;
3      $root \leftarrow$ overall utility ;
       // initialize criteria set with overall utility
4      $C \leftarrow root$ ;
5      while ($C \not\subseteq M$) do
6          $c \in C \setminus M$ ;
7          inquiry on $E$ to decompose $c$ into several sub-criteria $a$ ;
8          $C \leftarrow C \setminus c$ ;
9          $C \leftarrow C \cup a$ ;
10         create a new aggregation operator $n$ at position of $c$ ;
11         mark position of sub-criteria $a$ as children of $n$ ;
12         insert $n$ to $F$ as parent node of $a$;
13     end
14     insert all $c \in C$ into $F$ as leaf nodes ;
15 end
```

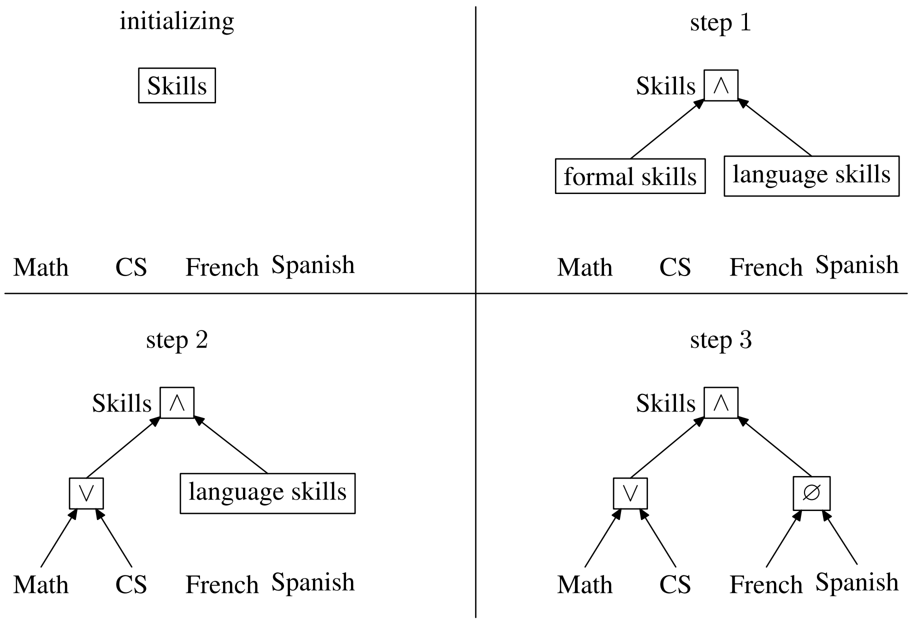


Figure 2.16: Interior structure elicitation in a top-down manner

Table 2.3: Attributes and related fuzzy sets for the car selection problem

Nr.	Attribute name	Description	Fuzzy set
1	“Buying”	the purchase price	*Buying*
2	“Maintain”	the maintaining cost	*Maintain*
3	“Doors”	number of doors	*Doors*
4	“Persons”	number of persons fit in the car	*Persons*
5	“Luggage”	size of the luggage boot	*Luggage*
6	“Safety”	the safety factor	*Safety*

relationship, which is finally expressed in the form of the newly created aggregation operator in the last column.

Table 2.4: Interior structure elicitation for car selection problem

Step	Candidate set	Aggregation	New operator
1	$\{Buying, Maintain, Doors, Persons, Luggage, Safety\}$	$Buying, Maintain \Rightarrow Price$	t-norm (“OR")
2	$\{Price, Doors, Persons, Luggage, Safety\}$	$Doors, Person, Luggage \Rightarrow Comfort$	t-conorm (“AND")
3	$\{Price, Comfort, Safety\}$	$Comfort, Safety \Rightarrow Technique$	OWA (“Average")
4	$\{Price, Technique\}$	$Price, Technique \Rightarrow Car$	t-norm (“AND")

This process is illustrated in Figure 2.17, where the red number in the braces of interior nodes indicates in which step the node (and its associated aggregation operator) is created.

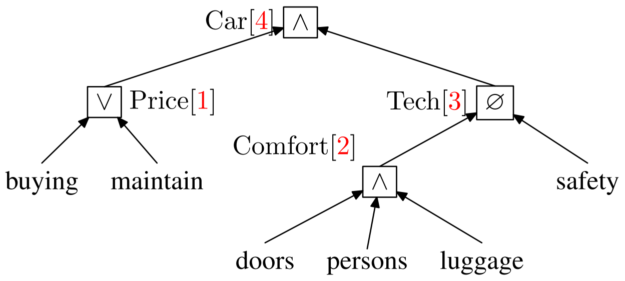


Figure 2.17: Interior structure elicitation for the car selection problem

Which method should be applied in practice, is again problem dependent. One can even combine these both methods in a way that one starts from top and bottom sides, and tries to decompose (from the top side) and aggregate (from the bottom side) criteria until a full FOT model is built.

2.5 Conclusions

In this chapter we introduced basic components to construct an FOT for modeling utility functions, as well as the definition and properties of FOT models. Several elicitation methods were proposed to help human experts to express the prior knowledge in the form of FOTs. To this end, the user is able to model a utility function represented by an FOT to evaluate a set of objects, entities or alternatives in an intuitive and convenient way.

Using an FOT to model a utility function takes benefit from its excellent interpretability, in which the whole FOT can be interpreted in natural language, such that the human being can understand an FOT perfectly. However there are still several untouched issues related to modeling a utility function with an FOT model, for instance, how to specify the parameters involved in FOTs precisely, if the human expert can not do this? Some of these issues are discussed in the following chapters.

3 Calibration of Fuzzy Operator Trees

As a utility function, an FOT model consists of two parts: A qualitative part and a quantitative one. The former part includes the tree-like structure of an FOT associated with fuzzy aggregation operators, linguistic hedges and the type of fuzzy sets, whereas the latter part corresponds to the parameter specification of an FOT, namely the specification of parameters involved in aggregation operators as well as linguistic hedges and fuzzy sets at the leaf nodes of an FOT.

Considering the applications of FOTs, let us distinguish following three cases according to the qualitative and quantitative parts:

- **Both parts can be specified** : In many cases, a human expert is able to provide both parts in detail, in which not only the structure of an FOT can be determined by human expert, but also the involved parameters can also be specified precisely. For example, in many industrial areas, human expert can provide the structure of FOTs based on the prior knowledge. On the other hand, there are the detailed specification sheets usually exist, which can be used to determine quantitative part of FOTs.

- **Only qualitative part can be specified** : In other situations, a human expert may still be able to give the qualitative structure of the model, but may have problems to specify the parameters precisely. For example, in many cases, specification sheets are not available, or it is impossible to specify an FOT based on them.

- **Both parts can not be specified** : There are still some cases where the qualitative structure of the model can not be extracted from human experts due to different reasons, for example, the quality assessment can not be modeled in the form of an FOT, or it is difficult to communicate with human expert, etc. In this case, one may question whether an FOT is a reasonable approach at all.

We address the second situation in this chapter. With the help of human experts, we can build an FOT representing the prior knowledge of experts as introduced in the previous section. Despite the good interpretability of an FOT, it does usually lack of accuracy, where disagreement is shown when being applied in practice, for example different results come from human experts and corresponding FOTs on exemplary data provided by human experts or extracted from historical records.

The basic idea in this chapter is to try to find an optimal parameter specification in a special *data-driven* way. In order to find parameters that lead to an optimal fit of the model, we try to tune the involved parameters to fit given exemplary data, called *calibration*, since it does not change the structure of FOTs during the calibration process. As an example for our calibration process, let us reconsider the problem of assessing candidates illustrated in Section 1.2.1. On the one hand, we can model the evaluation process of experts in the form of an FOT using elicitation techniques. On the other hand, a set of exemplary rating of concrete candidates is given by a human expert, which can be used to determine the parameters in the FOT. In this context, the above problem can be understood as follows: Knowing that the expert makes use of a qualitative evaluation scheme represented in the form of FOTs, how to find parameters such that the entire model imitates the expert in an optimal way. Note that the calibration based on given exemplary data becomes a typical *supervised learning* task in the field of machine learning [Mit97, HMS02]. The problem of calibration concerns to determine optimal parameters, so that the FOT can fit the exemplary data.

This chapter is organized as follows: In Section 3.1, we state the problem of calibration of FOTs formally. In Section 3.2, we discuss the identifiability of FOTs. The considered techniques for the purpose of calibration of FOTs are discussed in Section 3.3, and the experimental results are shown in Section 3.4. Section 3.5 concludes this chapter.

3.1 Problem Statement

Formally we can express an FOT F as a parametrized function:

$$F(x, \theta),$$

where x is the input feature vector, $\theta = [r_1, r_2, \ldots, r_k]$ is a vector of all involved parameters in F, r_i $(1 \leq i \leq k)$ denotes a parameter coming from an aggregation operator, linguistic hedge or fuzzy set.

Furthermore a set of given exemplary data (also called *training data*) with N tuples is expressed in the form of:

$$D := \{(x_i, y_i) | 1 \leq i \leq N\},$$

where x_i corresponds to an input feature vector for F, and y_i is the utility suggested by the expert according to x_i. The y_i is called *real output* related to x_i, since it is an output determined by human expert, while the overall utility deduced by applying F is called *estimated output* denoted as $\hat{y}_i$ (=$F(x_i, \theta)$) to distinguish it from the real output y_i.

Therefore the objective of calibration of FOT can be expressed in terms of the accuracy on a given training data in the following form:

$$f(\theta) := \sum_{i=1}^{N} \gamma(\hat{y}_i, y_i) \tag{3.1.1}$$

where $\hat{y}_i = F(x_i, \theta)$ is the estimated output of F under parameter specification θ and input vector x_i, γ is a suitable *loss function*, which gives positive penalties according the disagreement between estimated and real outputs, γ is also called *error function*. A loss function has generally following properties:

- $\gamma(\hat{y}, y) = 0$, if $\hat{y} = y$.
  No penalty is given by a "perfect" assessment.
- $0 \leq \gamma(\hat{y}_1, y) \leq \gamma(\hat{y}_2, y)$, if $\hat{y}_2 \leq \hat{y}_1 \leq y$, or $\hat{y}_2 \geq \hat{y}_1 \geq y$.
  The more distance to the real output, the more penalty would be assigned.

In the case of continuous outputs $\hat{y}_i \in [0,1]$, the squared error is selected here as a default loss function, and it is defined as:

$$\gamma(\hat{y}_i, y_i) := (\hat{y}_i - y_i)^2 \tag{3.1.2}$$

which is a common choice for regression problems.

In many applications, one would prefer a discrete or ordinal output rather than a continuous one, for example [WCW+08]. In that case, the discrete outputs can be converted from continuous ones using a discretizer (see Section 2.3.3), again we have several candidate loss functions, the simplest function is the so-called 0/1-loss, that is,

$$\gamma(\hat{y}_i, y_i) = \begin{cases} 0 & \text{if } \hat{y}_i = y_i \\ 1 & \text{otherwise} \end{cases}$$

However in the case of ordinal outputs, this becomes a problem of ordinal classification, in which the simple 0/1-loss neglects the order between discrete outputs. For example, assume that there is an ordinal output set with three categories (*small*,*middle* and *big*) and the real output is *big*, the penalties for two estimated outputs ($\hat{y}_1 =$*small* and $\hat{y}_2 =$*middle*) are both 1 under the 0/1-loss function. But $\hat{y}_2$ is arguably a better prediction than $\hat{y}_1$, although both are incorrect outputs. The problem in ordinal classification is that deviations from the real output are mostly incomparable, and this makes it difficult to define a reasonable loss function. For example in the previous case $\hat{y}_1 =$*small* and $\hat{y}_2 =$*big* are incomparable, if the real output is *middle*. To avoid this problem, we employ an alternative loss function, which is proposed in [Ren05] and denoted as *all-threshold* loss function: Let $0 = t_0 < t_1 < \ldots < t_k = 1$ denote the thresholds used for converting a continuous prediction $\hat{y} \in [0,1]$ into a discrete one $\mu \in \{1, \ldots, k\}$. If the sought (discrete) output is $y = j$, then the all-threshold loss is defined by:

$$\gamma(\hat{y}, y) = \begin{cases} 0 & \text{if } t_{j-1} \leq \hat{y} \leq t_j \\ \sum_{h=i}^{j-1} (t_h - \hat{y}) & \text{if } t_{i-1} \leq \hat{y} < t_i \leq t_{j-1} \\ \sum_{h=j}^{i} (\hat{y} - t_h) & \text{if } t_j \leq t_i < \hat{y} \leq t_{i+1} \end{cases} \tag{3.1.3}$$

According to the all-threshold loss function, the predicted output would not be penalized, if it lies in a correct interval, namely $[t_{j-1}, t_j]$; otherwise the loss is defined as accumulative distance to each thresholds between $\hat{y}$ and t_{j-1} or t_j. Figure 3.1 gives an example for $y = 4$. Since the output of FOTs is numeric, the all-threshold loss function is applied in this thesis by default.

As a loss function assigns a positive number to a parameter specification, the objective function f of FOT builds a mapping from the parameter space Θ to the real domain $\mathbb{R}^+$:

$$f : \Theta \rightarrow \mathbb{R}^+$$

Since the objection function is associated with a loss function, a parameter specification θ_1 is better than another parameter specification θ_2, if $f(\theta_1) < f(\theta_2)$. The goal of calibration of FOTs is to find an optimal parameter specification.

Definition 3.1 (Global Optimum)**.** A parameter specification $\theta^* \in \Theta$ is called *global optimum*, if it holds

$$\forall \theta \in \Theta : f(\theta^*) \leq f(\theta)$$

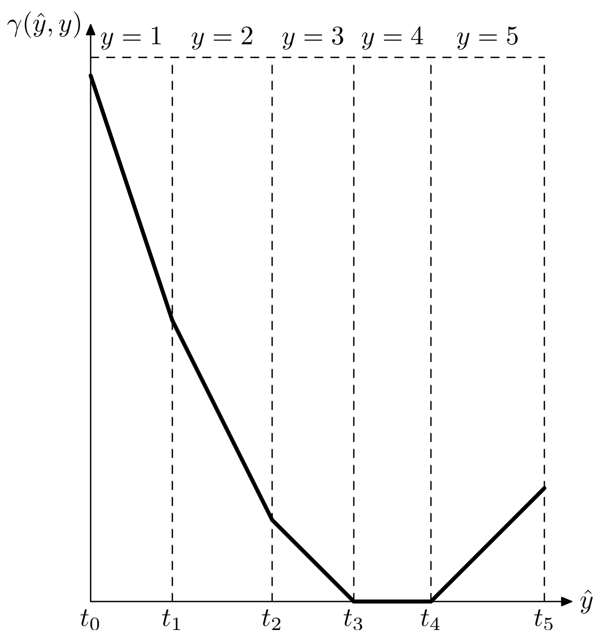


Figure 3.1: All-threshold loss function with real output $y = 4$

where Θ is the parameter space and f is the objective function of an FOT.

Definition 3.2 (Local Optimum)**.** A parameter specification $\theta' \in \Theta$ is called *local optimum*, if it holds

$$\exists \varepsilon > 0, \forall \theta \in \Theta : ||\theta - \theta'|| < \varepsilon \Rightarrow f(\theta') \leq f(\theta)$$

where ε is a positive number, which defines the area "near by" θ' and $||\theta - \theta'||$ calculates the Euclidean distance between two parameter specifications θ and θ'.

A global optimum indicates a θ, which gives the minimum objective value in $\mathbb{R}^+$, whereas a local optimum indicates a minimum among its neighbors. Obviously a global optimum is a local optimum at the same time. Usually for an optimization problem there are many local optima and global optima. For instance, Figure 3.2 shows the objective function for a one-dimensional θ, where θ^* denotes the position of global optimum and θ' of local optimum. A common problem is how to detect whether a θ is a global or local optimum, since the parameter space Θ is mostly too huge to make an exhaustive search. Many heuristic and stochastic approaches have been proposed to address this problem for different applications. We shall discuss several techniques regarding to calibration of FOTs later.

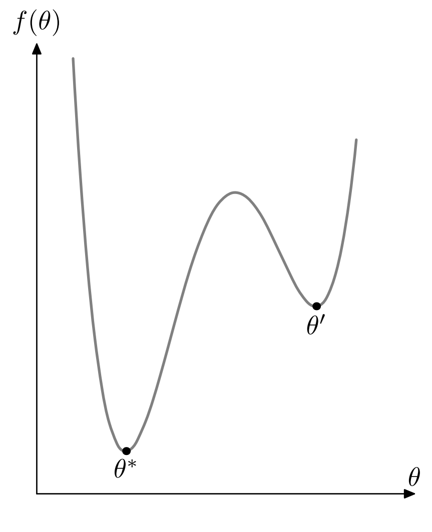


Figure 3.2: Global vs. local optimum

Note that the calibration problem of FOTs is a highly *non-linear* optimization problem, due to the fact that an FOT contains different types of operators, which are combined in a cascaded way. Moreover, it is a *constrained* optimization problem, since not all parameters can be chosen freely. For instance, if the model involves a modulating fuzzy set with trapezoid membership function, represented by its support $]l,h[$ and its core $[m,n]$, that is, this membership function can be learnt during the calibration process, then $l < m \leq n < h$ must be guaranteed.

3.2 Identifiability of Fuzzy Operator Tree

Before we go into the details of calibration techniques, we give some discussions about the identifiability of FOTs in this section. To this end, we can show that an FOT is not identifiable for the general case, but only under some restrictions.

Definition 3.3 (Identifiable)**.** A function $f(x,\theta)$ with inputs x and parameters θ is *identifiable*, if and only if

$$\forall\theta_1,\theta_2 \in \Theta : \theta_1 \neq \theta_2 \Rightarrow \exists x \in \mathbb{X} : f(x,\theta_1) \neq f(x,\theta_2) \tag{I1}$$

or

$$\forall\theta_1,\theta_2 \in \Theta : [\forall x \in \mathbb{X} : f(x,\theta_1) = f(x,\theta_2)] \Rightarrow \theta_1 = \theta_2 \tag{I2}$$

where Θ denotes the domain of parameter θ, and $\mathbb{X}$ is the domain of x.

The identifiability of a function depicts that a function is distinguishable as specified with different parameters. As the parameters of an identifiable function change, its behavior in terms of a mapping from inputs to outputs will change respectively. For identifiable functions, it is theoretically impossible that two different specifications have identical behavior.

As preparations for the identifiability theorem of FOTs, we show the identifiabilities of the components used to construct an FOT in this thesis at first.

- For the triangle and trapezoid membership functions, we have

  Proposition 3.4. *All triangle or trapezoid membership functions are identifiable.*

- For the Hamacher t-norms and t-conorms, we have

  Proposition 3.5. *The Hamacher t-norm or t-conorm is identifiable.*

- For the linguistic hedges, we have

  Proposition 3.6. *All linguistic hedges are identifiable.*

The proofs for these propositions are trivial and omitted here.

- For the OWA mixnorms, we have

  Proposition 3.7. *All OWA mixnorms are identifiable.*

  Proof. For any OWA mixnorm $M(x,\theta)$, where the input x (with dimensionality n) is represented by a feature vector: $x = [x_1,x_2,\ldots,x_n] \in [0,1]^n$ and $\theta = [w_1,w_2,\ldots,w_n]$, we try to deduce the contradiction according to the necessary condition of identifiability I1, namely if $M(x,\theta)$ is not identifiable, then it must exist two different parameter specifications for $M(x,\theta)$, say θ_1 and θ_2 ($\theta_1 \neq \theta_2$), so that for all $x \in [0,1]^n$, it holds $M(x,\theta_1) = M(x,\theta_2)$.

We shall deduce the contradiction of this assumption by showing that $\theta_1 = \theta_2$ to satisfy the previous condition. Without loss of generality, let us compare the first different element in θ_1 and θ_2, its position, say i, can be defined by:

$$i := \min\{j | j \in \{1,\dots,n\}, w_{1j} \neq w_{2j}, w_{1k} = w_{2k}, \forall 1 \leq k < j\}$$

An input x can be assigned in following form:

$$x = [\underbrace{0,0,\dots,0}_{n-i \text{ times}}, \underbrace{1,1,\dots,1}_{i \text{ times}}] \in [0,1]^n$$

then we have:

$$\begin{aligned} & M(x,\theta_1) = M(x,\theta_2) \\ \Rightarrow\ & \textstyle\sum_{j=1}^{n} w_{1j}\cdot y_j = \sum_{j=1}^{n} w_{2j}\cdot y_j \\ \Rightarrow\ & \textstyle\sum_{j=1}^{i} w_{1j} = \sum_{j=1}^{i} w_{2j} \end{aligned}$$

Together with the condition of $w_{1j} = w_{2j}$, for all $1 \leq j < i$, one can follow immediately that $w_{1i} = w_{2i}$. In this way, we can prove that $w_{1j} = w_{2j}$, for all $1 \leq j \leq n$, namely $\theta_1 = \theta_2$, which is contradictory to the assumption before. So we follow that any OWA mixnorm is identifiable.

□

Definition 3.8 (Completeness of FOT)**.** An FOT $F(x,\theta)$ is *complete* on the unit interval, if

$$\forall c \in [0,1], \forall \theta \in \Theta, \exists x \in \mathbb{X} : F(x,\theta) = c$$

For convenience of use, we denote an input, which lead to a fixed output of $F(x,\theta)$, as:

$$x^{=c} \in \{x \in X | F(x,\theta) = c\}$$

for any $\theta \in \Theta$ and $c \in [0,1]$, analog we have:

$$x^{>c} \in \{x \in X | F(x,\theta) > c\}$$

$$x^{<c} \in \{x \in X | F(x,\theta) < c\}$$

In order to show the identifiability of FOT, we make following assumptions:

- **Independence** : The independence of input variables involved in an FOT is assumed here. Consequently this requires that fuzzy sets do not share any input variable, because otherwise it is easy to see that basic evaluations based on shared input variables are not independent any more. Figure 3.3 shows a

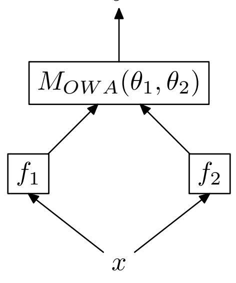


Figure 3.3: An unidentifiable FOT due to violation of independence

simple case of unidentifiable FOT due to violation of independence, where all involved linguistic hedges are specified with constant 1, two fuzzy sets (f_1, f_2) are identical, which share one input variable x. In this case, one can easily see that:

$$y = f(x)$$

for any parameter specification in M_{OWA}, so that the FOT expressed in Figure 3.3 is not identifiable.

- **Completeness** : Although as mentioned before that an FOT is able to deal with numeric and discrete inputs and outputs, we assume in this part that an FOT works only on numeric inputs, which are modulated by triangle or trapezoid membership functions as standard case. Furthermore the completeness of FOTs is assumed, in case of using trapezoid or triangle membership functions, this can be easily fulfilled by requiring the completeness of all membership functions, for example, considering a trapezoid membership function $A(x;l,m,n,h)$, the completeness of A requires the domain of x contains $[l,m]$ or $[n,h]$. Obviously, the completeness of all membership functions leads to the completeness of an FOT, as well as the completeness of all its sub-trees.

- **Constraints on linguistic hedges** : Furthermore, we assume that linguistic hedges involved in an FOT are fixed here, that is, we only consider different parameters in fuzzy sets and operators, whereas the parameters of linguistic hedges are treated as constants. This is an important condition for identifiability, otherwise the identifiability can be violated easily, for example, Figure 3.4 gives a simple case, where both FOTs only differ in three labeled linguistic hedges, the output of the subtree containing these linguistic hedges is $(\max(f_i(x_i), f_j(x_j)))^2$ in both FOTs (note that $M_{OWA}(1,0) = \max$).

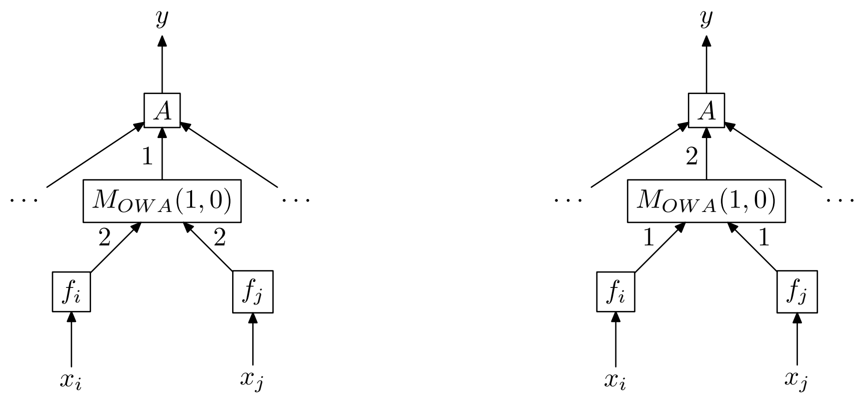


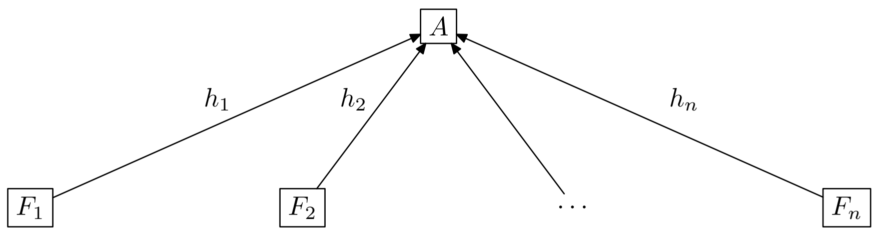
Figure 3.4: An unidentifiable FOT without constraint on linguistic hedges

Under the previous assumptions, we can show the identifiability of FOT:

Theorem 3.9. *An FOT satisfying the previous properties (independence, completeness and constraint on linguistic hedges) is identifiable.*

Proof. An FOT model can be constructed in a bottom-up way, just like Figure 3.5 shows, where A denotes an aggregation operator consisting of the Hamacher t-norm, t-conorm or OWA mixnorm. $F_1, \ldots, F_n$ are sub-trees of A, which are either fuzzy sets or FOTs. The edge connecting A and a sub-tree F_i can be associated with a linguistic hedge ($h_1, \ldots, h_n$ here). The idea of proof here is to show the identifiability of FOTs inductively, in which the identifiability of FOTs can be guaranteed if all its sub-trees and linguistic hedges are identifiable. To accomplish this, we try to examine the parameters in A, h_i or F_i for $i \in \{1, \ldots, n\}$ separately by applying the idempotency of aggregation operators as well as the linguistic hedges.

Figure 3.5: Constructing an FOT recursively



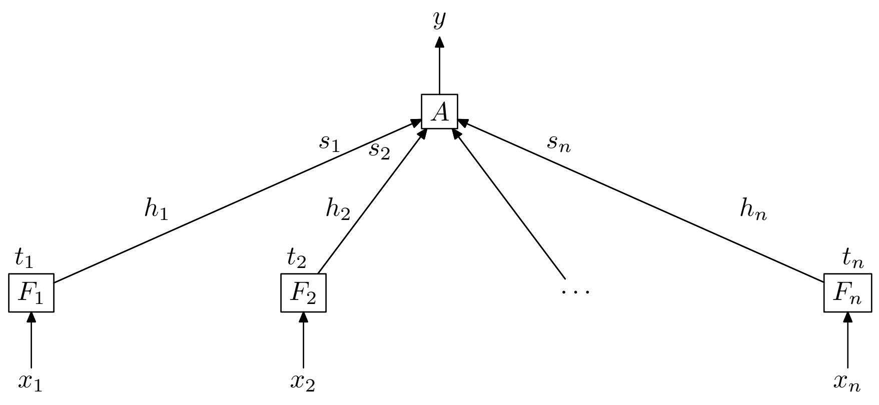
Note that having a closer look at the assumptions we have made before, it is not difficult to figure out that for any considered FOT here, the number of involved fuzzy sets equals the number of input variables and each input variable is numeric, since any input variable is associated with a triangular or trapezoid membership function independently. Moreover, the input x of an FOT can be expressed in the form of a feature vector of dimensionality m, $x \in \mathbb{R}^m$, if there are exactly m input variables involved.

Following, we give the proof using induction upon the aforementioned recursive structure:

- Basis step:
  At the lowest level of an FOT, the input variables are firstly modulated by fuzzy sets, which can be viewed as lowest sub-trees to construct an FOT. Since all fuzzy sets used in an FOT are triangle or trapezoid membership functions, the sub-trees at the lowest level of an FOT are identifiable.

- Induction step:
  Assume that all its sub-trees are identifiable, we intend to prove the identifiability of an FOT, which is constructed recursively. For the sake of clearness, Figure 3.6 illustrates such an FOT, say F, in detail, where y is the overall output of F, $x_i \in \mathbb{R}^{m_i}$ is input for the i-th sub-trees F_i (assume that m_i input variables involved in F_i) and t_i denotes the interior outputs of F_i, while s_i is output of t_i through the i-th linguistic hedge h_i, for all $i \in \{1,\ldots,n\}$.

Figure 3.6: Constructing an FOT recursively in detail



Again let us assume that F is not identifiable. Then, there must exist at least two parameter specifications, say θ_1 and θ_2, so that $\theta_1 \neq \theta_2$ and

$$\forall x \in \mathbb{R}^{m_1} \times \mathbb{R}^{m_2} \times \ldots \times \mathbb{R}^{m_n} : F(x,\theta_1) = F(x,\theta_2) = y$$

according to the necessary condition of identifiability I1. We try to prove that the parameters of θ_1 and θ_2 must be identical to fulfill the second condition. According to the type of aggregation operators, the following three cases are distinguished here:

1. if the aggregation operator A is associated with a Hamacher t-norm, let us have a closer look on the parameters in F. We shall consider the parameters in each sub-tree at first, and then the parameters in the Hamacher t-norm.
   - Without loss of generality, a sub-tree F_k is considered here for $k \in \{1,\ldots,n\}$. To suppress the influence of other sub-trees, we try to construct a special input, so that the overall output y only depends on the parameters in F_k and h_k, but is independent of the parameters in other sub-trees and A. Notice that 1 is the neutral element for t-norms, the criterion to construct the special input vector is to let the interior outputs of other sub-trees be 1, namely $t_m = 1$, for all $m \neq k$. From the completeness of FOT, the existence of such input vector is guaranteed.
     Following this idea, an input vector x' is selected in the following specific form:
     $$x' = [\overset{=1}{x_1}, \ldots, \overset{=1}{x_{k-1}}, x_k, \overset{=1}{x_{k+1}}, \ldots, \overset{=1}{x_n}]$$
     namely all sub-trees except F_k are assigned with an input vector, so that all their interior outputs are one. Due to the completeness assumption, this kind of input vector exists. So due to 1-idempotency of linguistic hedges, as well as the property of t-norm, we have:
     $$y = F(x') = A(1,\cdots,1,s_k,1,\cdots,1) = s_k = h_k(F_k(x_k)) \qquad (3.2.1)$$
     That is, for the specific input x', the overall output y equals s_k, which depends only on h_k and F_k. Since the parameter of linguistic hedges h_k is fixed, let us denote the parameters involved in F_k as θ_{1k} and θ_{2k} among the parameters in θ_1 and θ_2.
     Equation 3.2.1 can be extended as follows:
     $$\begin{array}{lll} & F(x',\theta_1) = F(x',\theta_2) & \forall x' \\ \Rightarrow & h_k(F_k(x_k,\theta_{1k}),\alpha_1) = h_k(F_k(x_k,\theta_{2k}),\alpha_2) & \forall x_k \\ \Rightarrow & F_k(x_k,\theta_{1k}) = F_k(x_k,\theta_{2k}) & \end{array}$$
     That is for any input of F_k under the parameter specifications θ_{1k} and θ_{2k}, the interior output of F_k are identical. Due to the fact that all sub-trees including F_k are conditioned identifiable and I2 from the definition of identifiability, we conclude that $\theta_{1k} = \theta_{2k}$.
   - Until now we have shown that the parameters in all sub-trees of θ_1 and θ_2 must be identical. Here we shall show the parameter in the root node (a Hamacher t-norm), say $r_1 \in \theta_1$ and $r_2 \in \theta_2$, must again be identical. Based on the previous conclusions, for any $x \in \mathbb{R}^m$, the same intermediate results $s_1,\ldots,s_n$ are returned under both parameter specifications θ_1 and θ_2, which are in turn the input values for the Hamacher t-norm in the root node, on the other hand the overall output y are identical under θ_1 and θ_2. From the identifiability of Hamacher t-norms, we can follow $r_1 = r_2$ immediately.
2. if the aggregation operator A is associated with a Hamacher t-conorm, the identifiability proof is very similar to that of a Hamacher t-norm. The only difference lies in the proof of the identity of parameters in sub-trees, where the input vector x is selected in following specific form:
   $$x' = [\overset{=0}{x_1}, \ldots, \overset{=0}{x_{k-1}}, x_k, \overset{=0}{x_{k+1}}, \ldots, \overset{=0}{x_n}]$$

namely all sub-trees are assigned with special input vectors, so that the interior output of sub-trees are all 0 except the selected sub-tree F_k. In this way, we can again suppress the influence of all other sub-trees and the Hamacher t-conorm.

3. otherwise the aggregation operator A must be an OWA mixnorm, say M, this time we shall check the parameters in the root node at first, then in the sub-trees.

   – Let us assume M has n parameters $(w_1, w_2, \ldots, w_n)$ and the i-th parameter involved in the root node from θ_1 and θ_2 is denoted as w_{1i} and w_{2i} respectively, just as to prove the identifiability of an OWA mixnorm. Let us denote the index $i \in \{1, \ldots, n\}$, so that $w_{1i} \neq w_{2i}$ and $w_{1j} = w_{2j}$, for all $1 \leq j < i$. An input x can be assigned in following form due to the completeness of FOT:

   $$x = [x_1^{=1}, x_2^{=1}, \ldots, x_i^{=1}, x_{i+1}^{=0}, \ldots, x_n^{=0}],$$

   namely, by assigning special input vector x there are exact i sub-trees, whose interior outputs are 1, while other sub-trees with interior outputs 0. Due to the 1(0)-idempotency of linguistic hedges, we have

   $$s_m = \begin{cases} 1 & \forall m \in \{1, \ldots, i\} \\ 0 & \forall m \in \{i+1, \ldots, n\} \end{cases}$$

   which are in turn the input values for the OWA mixnorm in the root node. This leads to:

   $$\begin{aligned} y = F(x, \theta) &= W(s_1, \ldots, s_n; w_1, \ldots, w_n) \\ &= W(1, \ldots, 1, 0, \ldots, 0; w_1, \ldots, w_n) \\ &= \sum_{m=1}^{i} w_m \end{aligned}$$

   Considering the condition $w_{1j} = w_{2j}$, for all $1 \leq j < i$, we follow that $w_{1i} = w_{2i}$, consequently all parameters in the OWA mixnorm must be identical under the parameter specifications θ_1 and θ_2.

   – For any sub-tree, say F_k without loss of generality, and the linguistic hedge h_k $(1 \leq k \leq n)$ respectively, we want to design a specific form of input vector x again like in the previous cases, to suppress the influence of other sub-trees. From the definition of the OWA mixnorm, one can conclude that, for an OWA mixnorm with n parameters $([w_1, \ldots, w_n])$:

   $$\exists p \in \{1, \ldots, n\} : w_p > 0$$

   According p, the input vector of x for this case is selected in the following form due to the completeness of FOT:

   $$x' = [x_1^{=J(1)}, \ldots, x_{k-1}^{=J(k-1)}, x_k, x_{k+1}^{=J(k+1)}, \ldots, x_n^{=J(n)}]$$

   where

   $$J(i) = \begin{cases} 1 & \text{if } k \geq p, i < p \\ 1 & \text{if } k < p, i \leq p \\ 0 & \text{otherwise} \end{cases}$$

for all $i \in \{1,\dots,n\}$, namely all sub-trees except F_k are assigned with special inputs, so that there are exactly $p-1$ sub-trees with interior outputs 1, and $(n-p)$ sub-trees with 0. Let us denote the parameters involved in F_k as θ_{1k} and θ_{2k}, and the parameter in h_k as α. So due to 1(0)-idempotency of FOT and linguistic hedge, we have:

$$\begin{aligned} y = F(x',\theta) &= W(s_1,\dots,s_n;w_1,\dots,w_n) \\ &= W(\underbrace{1,\dots,1}_{(p-1)\text{ times}},s_k,0,\dots,0;w_1,\dots,w_n) \\ &= \sum_{i=1}^{p-1} w_i + s_k \cdot w_p \end{aligned}$$

together with the previous conclusion follows that:

$$\begin{aligned} & F(x',\theta_1) = F(x',\theta_2) = y && \forall x' \\ \Rightarrow\quad & s_k(\theta_1) = s_k(\theta_2) && \forall X_k \\ \Rightarrow\quad & (F_k(X_k,\theta_{1k}))^\alpha = (F_k(X_k,\theta_{2k}))^\alpha \\ \Rightarrow\quad & F_k(X_k,\theta_{1k}) = F_k(X_k,\theta_{2k}) \end{aligned}$$

Then we get a similar condition as in the case of a Hamacher t-norm, which is based only on the k-th sub-tree (F_k) and the k-th linguistic hedge (h_k). Using the same technique we can deduce that

$$\theta_{1k} = \theta_{2k} \qquad \square$$

3.3 Calibration Techniques

As mentioned before, the goal of the calibration of FOTs is to find the global optimum θ^* for given training data, where $\theta^* \in \Theta$ in the parameter space of an FOT. Note that Θ is a real-valued constrained space, since the parameters of an FOT are real-valued and constrained respectively, either from aggregation operators, or from linguistic hedges or fuzzy sets. There are many techniques proposed in last decades [KHS01, GBNP96, Coe06, BBPV04] for this kind of optimization problems, and in practice no universally optimal method exists for all kind of problems. The optimization techniques can be categorized into following classes:

1. **Exhaustive search** : In the case of relatively small search space, it is possible to check through all potential solutions in the space, and to locate the global optimum precisely, also called *complete enumeration*. This kind of techniques becomes infeasible, as the search space becomes large, therefore it can be only applicable in small part of optimization problems. For the real-valued optimization problem it is usually not appropriate.

2. **Iterative search** : An iterative search usually starts with an initial solution, which can be generated, for instance, randomly. After that, one tries to make small change on a known solution using different approaches, in order to find a better solution. This process is repeated until some stop-criteria are reached. Since this search usually gets stuck in a local optimum, it is customary to carry out the process several times, starting with different initial solutions and saving the best result. We shall take a closer look at two representative approaches regarding to calibration of FOTs: Gradient descent (in Section 3.3.1) and simulated annealing (in Section 3.3.2).

3. **Population-based search** : In contrast to the iterative search, a population-based search, as the name indicates, maintains a population during the optimization process, which is a set of solutions. The solutions in population would be updated in various way, for instance, communicating with other solutions or making (small) changes, until no further improvement can be found. We are going to consider the evolutionary algorithms based on the principles of the evolution theory, which has shown an astonishing performance in numerous optimization problems. Section 3.3.3 is devoted for one of variants in the evolutionary algorithm, namely evolution strategies.

3.3.1 Gradient Descent

Gradient descent (**GD** for short) is an optimization algorithm, which can be used to find the local minimum of a function which presupposes that the gradient of the function can be computed [CK93, Jan93]. The key idea behind original gradient descent algorithm is as follows: If a real-valued function $f(x)$ is differentiable in a neighborhood of a point, say x_0, then $f(x)$ decreases fastest if one goes from x_0 in the direction of the negative gradient of $f(x)$ at x_0, denoted as $-\nabla f(x_0)$. It follows that $f(x_0) \geq f(x')$, where $x' = x_0 - \gamma \nabla f(x_0)$ for a positive, small enough number γ. The derivation of $f(x)$ at x_0 shows the direction of steepest ascent along the surface of $f(x)$, so a movement in the reverse direction will reach a lower position than $f(x_0)$, as long as this movement is small enough. The derivation of newly determined position can help moving to another lower place, until it lies already on a global or local minimum. The method of gradient descent, also known as the method of *steepest descent*, starts at an initial point, say x_0, after that one moves from x_i to x_{i+1} iteratively by making small changes in the direction of the negative gradient of $f(x)$ at x_i, until any further move can only increase $f(x)$.

Regarding to calibration of FOTs, we consider the default loss function (mean squared error) based on the parameter space Θ, where any parameter specification $\theta \in \Theta$ is a vector of parameter values involved in an FOT. One important reason to use this loss function is that it is differentiable at any parameter specification $\theta \in \Theta$. On the other hand, the fuzzy operators in FOTs, as well as the membership functions and linguistic hedges, are differentiable at any valid parameter values, so that an FOT, which combines these in a cascaded way, is again differentiable at any parameter specification. So the gradient descent method can be used here to find an optimal parameter specification.

Figure 3.7 illustrates the gradient descent algorithm, where the loss function $f(\theta)$ is considered based on an FOT with two parameter variables (θ_1 and θ_2). Given a parameter specification θ_0, the derivation of $f(\theta)$ at θ_0 is marked to show the steepest ascent along the surface of f. In order to move to a neighbor of θ_0 with lower loss value, both parameters θ_1 and θ_2 are tuned by changing their values in the reverse direction of the partial derivation of $f(\theta)$ on θ_1 and θ_2 respectively, as the solution θ_0' indicates.

Formally we take an FOT $F(x;\theta)$ based on input vector x and parameter vector θ, where $\theta = [\theta_1, \ldots, \theta_n]$ and there is training data D with tuples in the form of (x, y). For a parameter specification θ, the loss function is defined as:

$$\begin{aligned} f(\theta) &= \sum_{(x,y)\in D} \gamma(\hat{y}_i, y_i) \\ &= \sum_{(x,y)\in D} (F(x;\theta) - y)^2 \end{aligned} \tag{3.3.1}$$

Indeed $f(\theta)$ depends on both θ and tuples (x, y) in D, but since we assume the train-

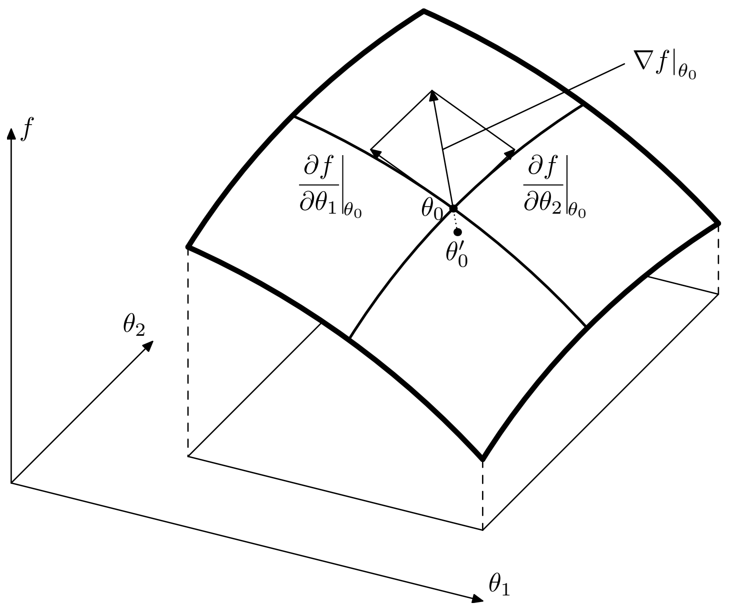

$\nabla f|_{\theta_0}$
f
$\frac{\partial f}{\partial \theta_1}\Big|_{\theta_0}$
θ_0
$\frac{\partial f}{\partial \theta_2}\Big|_{\theta_0}$
θ_0'
θ_2
θ_1

Figure 3.7: Visualization of gradient descent

ing data are fixed during the optimization process, $f(\theta)$ can be seen as a function on the parameter vector θ. To determine the direction of gradient descent, the derivation of $f(\theta)$ is computed with respect to each component of the parameter vector θ, we call the derivations the *gradient* of the loss function f on θ, written $\nabla f(\theta)$. We have:

$$\nabla f(\theta) = [\frac{\partial f}{\partial \theta_1}, \frac{\partial f}{\partial \theta_2}, \dots, \frac{\partial f}{\partial \theta_n}]$$

Note that each component of $\nabla f(\theta)$ is a gradient of f with respect to one parameter in θ, whose positive value shows a small increase of related parameter leads to an increase of the overall loss value, a negative gradient indicates the other direction correspondingly. To reach a lower loss value, θ is updated in a following manner:

$$\theta \leftarrow \theta + \Delta\theta \tag{3.3.2}$$

where

$$\Delta\theta = -\eta \nabla f(\theta) \tag{3.3.3}$$

Adding a negative sign is because we want to change θ in a direction so that $f(\theta)$ decreases. Here η is a positive value called the *learning rate*, which determines the step size of change on θ. For each parameter in θ, say θ_i, the updating rule 3.3.2 can be written:

$$\theta_i \leftarrow \theta_i - \eta \nabla f(\theta_i) \tag{3.3.4}$$

which makes clear that the gradient descent is achieved by altering each component θ_i of θ in proportion to $\frac{\partial f}{\partial \theta_i}$.

The gradient of f on θ is, fortunately, not complicated, which can be obtained by differentiating f based on Equation 3.3.1, as:

$$\begin{aligned}
\nabla f(\theta_i) &= \frac{\partial f}{\partial \theta_i} \\
&= \frac{\partial}{\partial \theta_i} \sum_{(x,y)\in D} (F(x;\theta) - y)^2 \\
&= \sum_{(x,y)\in D} \frac{\partial}{\partial \theta_i} (F(x;\theta) - y)^2 \\
&= \sum_{(x,y)\in D} 2(F(x;\theta) - y) \frac{\partial}{\partial \theta_i} (F(x;\theta) - y) \\
&= \sum_{(x,y)\in D} 2(F(x;\theta) - y) \frac{\partial (F(x;\theta))}{\partial \theta_i}
\end{aligned} \tag{3.3.5}$$

on the i-th parameter of θ, where the derivatives of y is zero, since y are treated as constant in training data D. To calculate $\frac{\partial(F(x;\theta))}{\partial\theta_i}$, we consider following two cases:

- If θ_i is a parameter in the root node of F, its partial derivation can be computed directly.
- Otherwise, the partial derivative on non-root nodes can be calculated with the chain rule. Since this method can be applied in a recursive manner, we generalize this problem in Figure 3.8 for following discussion, where A denotes an aggregation operator, whose k-th input is built by A_k (an aggregation operator in turn or a fuzzy set) and h_k (a linguistic hedge), this method is applied to get the partial derivatives on h_k and A_k.

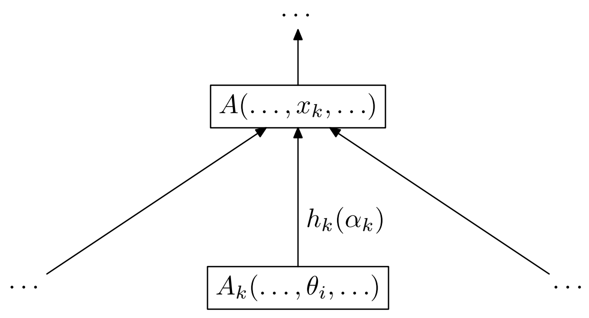


Figure 3.8: Compute derivatives using the chain rule

In FOTs the partial derivative on a parameter θ_i can be derived in a quite convenient way using the chain rule, let us distinguish the position of a parameter θ_i in following cases:

1. If it is a parameter involved in a linguistic hedge directly connected to A, the gradient of A on θ_i is:

$$\frac{\partial A}{\partial \theta_i} = \frac{\partial A}{\partial x_k} \cdot \frac{\partial x_k}{\partial \theta_i}$$

2. Otherwise it must be a parameter involved in a sub-tree of A, say A_k, the gradient of A on the parameter θ_i can be written as:

$$\frac{\partial A}{\partial \theta_i} = \frac{\partial A}{\partial x_k} \cdot \frac{\partial x_k}{\partial \theta_i} = \frac{\partial A}{\partial x_k} \cdot \frac{\partial h_k}{\partial x} \cdot \frac{\partial A_k}{\partial \theta_i}$$

Namely, in this case the partial derivative $\frac{\partial A}{\partial \theta_i}$ on A is related to the partial derivative $\frac{\partial A_k}{\partial \theta_i}$ on the sub-tree of A, this can be used in a recursive way.

The update rule for θ_i yields:

$$\Delta\theta_i = -\eta \sum_{(x,y)\in D} (F(x;\theta) - y) \cdot \frac{\partial F(x;\theta)}{\partial \theta_i} \tag{3.3.6}$$

where the constant factor 2 in Equation 3.3.5 is integrated into the learning rate η.

This method is referred as *direct gradient descent* ($GD - direct$ for short) in the following discussion.

Inspired by the well-known error back-propagation algorithm of optimization on neural networks, we notice that the loss function of FOTs measures errors by applying FOTs on training data essentially, so that it is possible to propagate errors backward in FOTs to update parameters, in order to decrease overall errors on training data. Here we consider another method for updating parameters in FOTs, which works in a similar way like the well-known error back-propagation algorithm [RHW86b].

Again the problem is generalized in Figure 3.9, where we assume that the error of an operator or linguistic hedge, say A, is e, and the k-th subtree of A is denoted as A_k. To estimate error on A_k , x_k is treated as a parameter, since x_k is the output of A_k. On x_k, we can use Equation 3.3.3 to derive an expected update on x_k, which is then treated as the estimated error on A_k, written as e_k here. For the root node of an FOT, its error is simply the output of the loss function, while for other operators or linguistic hedges involved in an FOT, its error can be estimated in the previous way recursively.

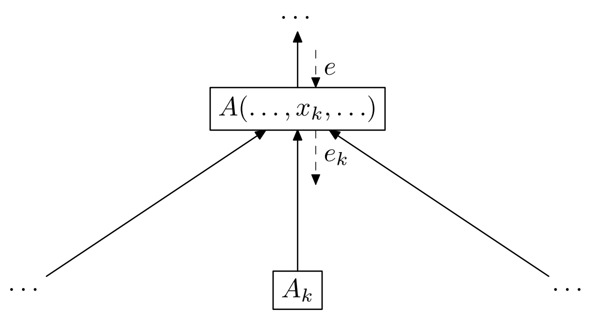


Figure 3.9: Error Backpropagation

Generally speaking, the update rule for any parameter θ_i using error back-propagation yields:

$$\Delta\theta_i = -\eta \sum_{(x,y)\in D} e \cdot \frac{\partial F_k(x;\theta)}{\partial \theta_i} \tag{3.3.7}$$

where the constant factor 2 is integrated into the learning rate η, and e is the (estimated) error on operators in FOTs, which contains θ_i. We shall name this method as *static gradient descent* (GD for short) in the experiments later.

The gradient descent algorithm updates the parameter θ according to the updating rule iteratively, in which the parameters are randomly initialized at first. Then for each tuple from the training data D, the $\Delta\theta$ is calculated depending on the updating rule in GD or $GD - direct$ and added into θ. The process is repeated until any change on θ increases the overall loss value (error). This procedure is summarized in Algorithm 3.1.

In line 5 all tuples from the training data D are input into an FOT, and the changes $\Delta\theta_i$ are summarized, then added to θ_i once in line 16. So every time scanning on D makes an update on θ, which is called *batch learning*. Another way to update θ is to add $\Delta\theta_i$ into θ_i after scanning each tuple in D immediately, called *online learning*, which can

Algorithm 3.1: Gradient descent algorithm

```
    Input: FOT $F(x;\theta)$, training data $D$, updating type $t$
 1  begin
 2      initialize each $\theta_i \in \theta$ randomly ;
 3      repeat
 4          initialize each $\Delta\theta_i$ to zero ;
 5          for (each tuple $(x,y) \in D$) do
 6              input $x$ to $F$ ;
 7              compute the estimated output $\hat{y}$ ;
 8              if ($t$ is GD) then
 9                  $\Delta\theta_i \leftarrow \Delta\theta_i - \eta\nabla e(\theta_i)$ recursively ;
10              else
11                  for (each $\theta_i \in \theta$) do
12                      $\Delta\theta_i \leftarrow \Delta\theta_i - \eta(\hat{y}-y)\cdot\frac{\partial F(x;\theta)}{\partial\theta_i}$
13                  end
14              end
15          end
16          for (each $\theta_i \in \theta$) do
17              $\theta_i \leftarrow \theta_i + \Delta\theta_i$
18          end
19      until ($f(\theta)$ increases) ;
20  end
```

be achieved with a modification on the previous algorithm by moving the *For* loop in line 16 one row ahead. Generally the batch learning provides a stable performance in contrast to the online one, whereas the later one is usually much faster.

One challenge regarding the gradient descent algorithm is to select a proper learning rate η. Theoretically a sufficiently small η is preferred to guarantee a real "descent" of loss value, but on the other hand a small η may cause to a slow learning speed, in which algorithm has to run a long time until the stop criterion is met. If the η is too large, the gradient descent search runs the risk of overstepping the optimum, or getting in an oscillation. Figure 3.10 demonstrates the possible effects of improper learning rate, where θ_0 indicates the initial solution, the dashed line shows the movement tracks under a too small or too large learning rate respectively. For this reason, one common modification to the algorithm is to gradually reduce the learning rate η as the number of gradient descent steps grows.

Many other variants of gradient descent have been invented to eliminate the problem with learning rate. Using a time-dependent learning rate, Equation 3.3.2 to 3.3.3 can be rewritten as:

$$\theta(t+1) = \theta(t) + \Delta\theta(t)$$
$$\Delta\theta(t) = -\eta(t)\nabla f(\theta(t))$$

with the iteration time t.

Most of these variants differ in the definition on previous two formulas, here we are considering several variants by listing their main differences:

1. **Manhattan training**

$$\Delta\theta(t) = -\eta(t) * sgn(\nabla f(\theta(t)))$$

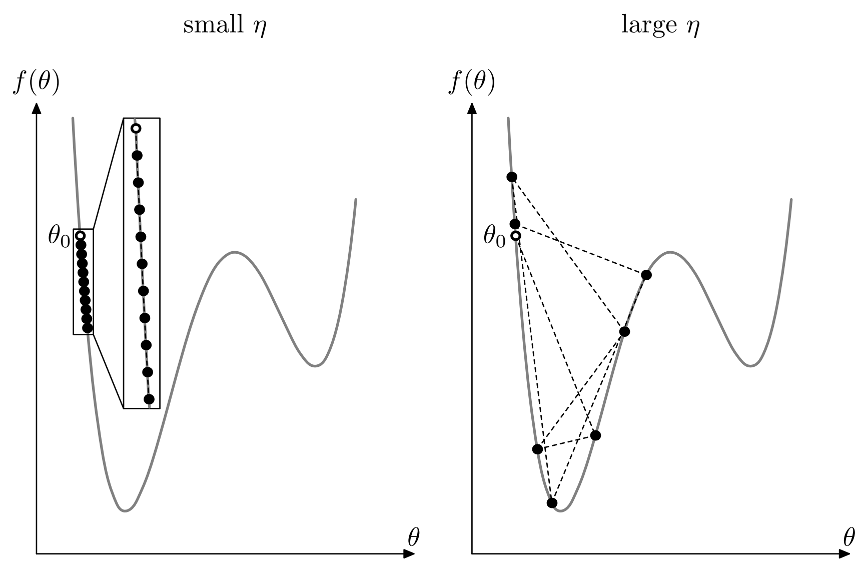


Figure 3.10: Effect of learning rate in gradient descent

where *sgn* is the signum function to extract the sign of gradient of f with respect to θ at t-th generation [BKKN03]. The numerical value of gradient does not play a role in the Manhattan training, but its sign, so that the update of parameters is only determined by direction of gradient and the learning rate (depend on the iteration time t).

2. **Momentum term**

$$\Delta\theta(t) = -\frac{\eta(t)}{2} * \nabla f(\theta(t)) + \beta * \Delta\theta(t-1)$$

The Momentum term [RHW86a, Sar95] considers not only the update of parameters determined by current gradient routinely, but also the update in the last iteration ($t-1$), in which a proportion of the previous change is added to every new change in θ. A common choice for additional parameter β lies in $[0.5, 0.95]$, in this thesis we shall take $\beta = 0.7$ as default setting.

3. **Self-adaptive error back-propagation**

$$\eta(t) = \begin{cases} c^- * \eta(t-1) & \text{, if } \nabla f(\theta(t)) * \nabla f(\theta(t-1)) < 0, \\ c^+ * \eta(t-1) & \text{, if } \nabla f(\theta(t)) * \nabla f(\theta(t-1)) > 0, \\ \eta(t-1) & \text{, otherwise.} \end{cases}$$

The idea of self-adaptive error back-propagation [Sar95] is to update the learning rate according to gradient at current and last iterations. As gradients at both iteration have same directions ($\nabla f(\theta(t)) * \nabla f(\theta(t-1)) > 0$), it means that optimum is still far away from current solution, the update would be increased with a "greater" learning rate, otherwise with a "smaller" one. The suggested domains for additional parameters c^- and c^+ are $[0.5, 0.7]$ and $[1.05, 1.2]$. A standard setting $c^- = 0.6$ and $c^+ = 1.1$ are employed in this thesis, as well for next variants.

4. **Resilient error back-propagation**

$$\Delta\theta(t) = \begin{cases} c^- * \Delta\theta(t-1) & \text{if } \nabla f(\theta(t)) * \nabla f(\theta(t-1)) < 0, \\ c^+ * \Delta\theta(t-1) & \text{if } \nabla f(\theta(t)) * \nabla f(\theta(t-1)) > 0 \\ & \text{ and } \nabla f(\theta(t-1)) * \nabla f(\theta(t-2)) \geq 0, \\ \Delta\theta(t-1) & \text{otherwise.} \end{cases}$$

The Resilient error back-propagation [RB94] takes gradients in current and last two iterations into account to increase or decrease the update on parameters.

5. **Quick-propagation**

$$\Delta\theta(t) = \frac{\nabla f(\theta(t))}{\nabla f(\theta(t-1)) - \nabla f(\theta(t))} * \Delta\theta(t-1)$$

The Quick-propagation [Fah88] considers gradients in current and last iterations, in which the ratio of gradient change is calculated.

Notice that it is proved the gradient descent can always converge to an optimum with a proper defined learning rate [Mit97], but not necessary to a global optimum. In a multimodal case, where many optima coexist, the gradient descent algorithm has a risk to get stuck in one local optimum. This problem can not be completely avoided in principle, but one can weaken it using different techniques, for instance restart training process with different initializations. We shall check its performance in the later experimental section.

3.3.2 Simulated Annealing

The gradient descent algorithm is a greedy search, in which only a change with improvement is allowed. Usually the success of a gradient descent algorithm is very dependent on the initial position, so if the initial position lies nearby a local optimum, it is not able to jump out of the optimum to search another optimum. A technique to avoid this problem is the so-called *simulated annealing* (SA for short) [KGV83, JAMS89, JAMS91, LA87, SPR04], which is inspired by annealing in metallurgy. The basic idea behind simulated annealing is to give a probability to accept a “worse” solution, the probability depends on, first of all, the objective value of solution and a global parameter (called *temperature*), which is gradually decreased during the process (called *temperature schedule*). As a result, the SA has relative high probability to accept a “worse” solution at the beginning in the process, and this probability decreases as the search progresses, so it tends to only accept a solution with improvement to the end of the search process.

Generally, higher temperatures correspond to a greater probability to accept a worse solution, so that large moves are allowed in the search space, while lower temperatures correspond to greater probability only allow to move to a local optimum, which corresponds small moves for improvement of current solutions. The idea of using simulated annealing is that the earlier (more random) moves have led the algorithm to the deepest “basin” in the objective function surface. In fact, one of the appeals of the approach is that it can be mathematically proved that the simulated annealing can always find a global optimum, if one is using the appropriate temperature schedule [Haj88, Gid85]. However, under this kind of temperature schedule, the temperature is decreased very slowly, until a solution is determined. Therefore, this kind of temperature schedule is usually not efficient enough to be used in practice.

Figure 3.11 shows an example visually based on one dimensional parameter θ and its objective function $f(\theta)$ indicated by the bold line, and θ_0, θ', θ^* are the initial solution, a local optimum, a global optimum respectively. As θ_0 starts with a high temperature, the probability stepping over θ' to θ^* becomes higher comparing a θ_0 with lower temperature.

Following the similar idea of gradient descent, the process of simulated annealing is shown in Algorithm 3.2. Firstly, a random solution p is selected in the search space, and its objective value is calculated with given training data. In the iteration step,

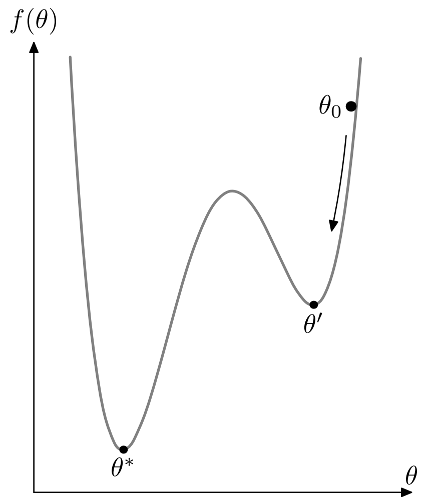


Figure 3.11: Simulated annealing

a neighbor of p is generated by adding a small change η, written as p'. According to the objective value of p' and the temperature parameter related to the iteration number i, we check whether to accept p' or not. The parameter η serves here just like in gradient descent as a learning rate, whose value is very important to success of search. To check the stop criterion in line 14, a probability is calculated based on the objective values and temperature parameter, where:

Δf the decrease of objective value from f to f', namely

$$\Delta f = f' - f$$

m the approximated domain of objective values, is defined as the maximal observed deviation of objective value,

$$m = \max(\Delta f), \text{ for all } \Delta f$$

T the temperature parameter, which decreases during iterations. A common choice is a decreasing function on the number of iteration i, for example:

$$T = \frac{1}{i}$$

3.3.3 Evolution Strategies

In computational intelligence, an evolutionary algorithm(*EA* for short) is a subset of evolutionary computation, a generic population-based meta-heuristic optimization algorithm, which is inspired by biological evolutionary theory [Wei02, Coe06, SR95, Sch93]. Some mechanisms from biological evolution are used in EA: mutation, recombination, natural selection and survival of fittest. During the last decades, the EA has gained increasingly importance in the field of computational intelligence, and becomes a powerful tool for the optimization problems [BS02]. Table 3.1 lists the commonly used terminologies taken from the evolutionary theory and their interpretations in the context of evolutionary algorithm.

There are currently four main techniques based on the principle of EA:

1. **Genetic algorithm** (*GA* for short) : It is a most popular type of EA, which is characterized by its representations in the form of number-strings [Rot06, Gol89]. Traditionally solutions are encoded in binary as strings of 0s and 1s,

Algorithm 3.2: Simulated annealing algorithm

```
    Input: FOT $F(x;\theta)$, training data $D$
    Output: Solution $p$
1   begin
2       generate a solution $p$ randomly ;
3       $i \leftarrow 0$ ;
4       evaluate $p$ with $D$ to get $f'$ ;
5       stop-flag ← false;
6       repeat
7           $f \leftarrow f'$ ;
8           $p' \leftarrow p + \eta$ ;
9           evaluate $p'$ with $D$ to get $f'$ ;
10          $i \leftarrow i + 1$ ;
            // check the stop criterion
11          if ($f' \leq f$) then    /* accept a not-worse solution */
12              $p \leftarrow p'$;
13          else /* otherwise it depends on a probability */
14              $q \leftarrow e^{-\frac{\Delta f}{m \cdot T}}$ ;
15              $r \leftarrow$ random number in $[0,1]$ ;
16              if ($r \leq q$) then /* with probability $q$ to accept a
                worse solution */
17                  $p \leftarrow p'$;
18              else        /* with probability $1-q$ to stop */
19                  stop-flag ← true ;
20              end
21          end
22      until (stop-flag) ;
23      return $p$
24  end
```

Table 3.1: Terminologies of evolutionary algorithm

Term	Explanation	Symbol
individual	candidate solution	I
generation	iteration of an evolutionary algorithm	t
population	a set of individuals in a generation	P
population size	the number of individuals in a population	μ
offspring (or child)	newly generated individuals in a generation	
parent	individuals, from which off-springs are generated	
parent selection	the technique to choose parent(s) to generate offspring(s)	
parent size	the number of selected parents to generate an offspring	ρ
off-springs size	the number of newly generated off-springs in a generation	λ
environment selection	the technique to construct a population for next generation	sel
fitness	the goodness of individual	f
recombination (or crossover)	the technique to create offspring(s) by exchanging information of parents	rec
mutation	the technique to create offspring(s) by change parent(s)	mut

which are particularly useful for solutions represented by discrete values. Genetic algorithm dealing with real-valued features has also been researched in several works [KD91, Gol91].

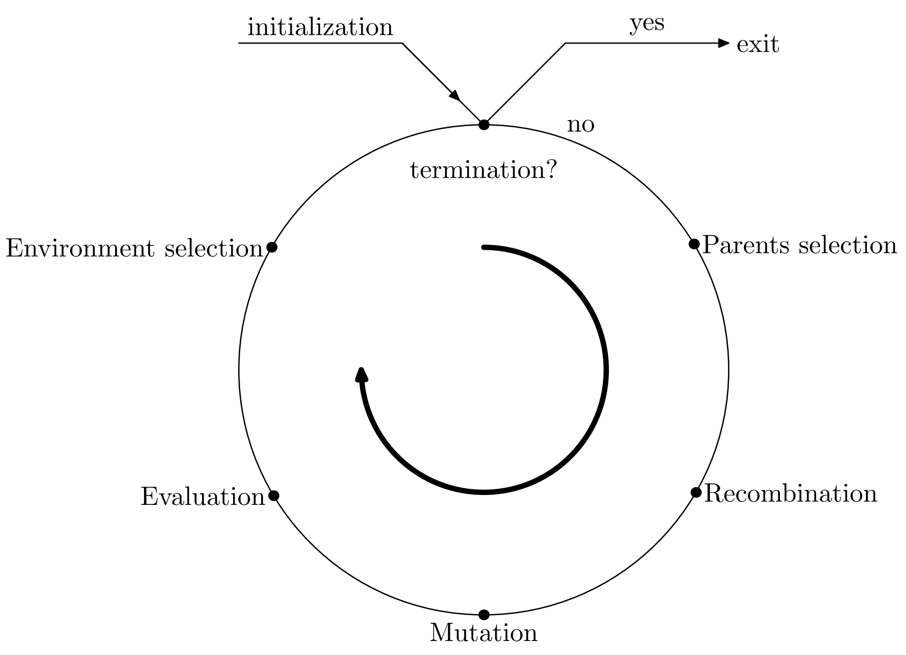
2. **Genetic programming** (*GP* for short) : This is a specialization of genetic algorithms where each individual is a computer program usually, traditionally an individual is represented as tree structures with strings [Koz92]. Originally GP favors the use of programming languages that naturally embody tree structures (for example, Lisp), non-tree representations have been also suggested and implemented successfully, such as in [BNKF01] .

3. **Evolutionary programming** (*EP* for short) : As emerged in 1960's, EP works just like genetic programming but with fixed structure , and can handle real-value numbers [YL96]. Currently the EP is extended to deal with flexible structure or representation, so it is becoming harder to distinguish from the next variant of EA, evolution strategies.

4. **Evolution strategies** (*ES* for short) : ES uses problem-dependent representations, and is especially desired for optimization problems with real-valued solutions [BS02]. For the purpose of calibration of FOTs, in which the main task is to optimize the continuous parameters, this technique is employed in this thesis. In the following, we concentrate on ES and introduce the corresponding terminologies of ES, as well as the application of ES for calibration of FOTs.

The ES, developed by Rechenberg and Schwefel at the Technical University of Berlin in the 1960s [Rec73], simulates the natural evolution process by executing the evolutionary loop illustrated in Figure 3.12. Like common in the evolutionary algorithm, an individual (standing for a candidate solution in the ES) is represented by a vector with real-valued numbers, where the initial step of ES constructs a population with finite number of individuals randomly. After the evaluation of individuals in the initial population, the algorithm enters a loop, in which the individuals are recombined, mutated, evaluated and finally selected according their objective values (fitness values), so long as the termination criteria are not fulfilled. Following we describe these steps in detail.

Figure 3.12: Workflow of evolution strategies



3.3.3.1 Representation

The individual, as usually in evolutionary algorithm, ought to represent a candidate solution for considered problem. In ES an individual consists of several components in the form as follows:

$$I(t) = \left(\underbrace{\theta_1^{(t)}, \ldots, \theta_n^{(t)}}_{\text{object component } \theta^{(t)}}, \underbrace{\sigma_1^{(t)}, \ldots, \sigma_n^{(t)}}_{\text{strategy component } \sigma^{(t)}}, f(\theta^{(t)}), K \right)$$
$$= (\theta^{(t)}, \sigma^{(t)} f(\theta^{(t)}), K)$$

in the generation t, where $\theta^{(t)}$ is called *object component* to represent a candidate solution. In the case of FOT, the object component is a vector of n parameters involved an FOT, which can be the parameters in operators, linguistic hedges or membership functions. As the structure of FOTs are treated fixed during the calibration process, any FOT with known structure can be identified with its parameter specifications. Note that in this thesis, these parameters are all real-valued numbers, but have different domains and constraints, which has to be considered in the implementation. Beside the actual solution in $\theta^{(t)}$, an additional *strategy component* ($\sigma^{(t)}$) of dimension n is attached in an individual, which is a vector of n real-valued numbers, each of them serves to control the stepsize of one parameter of the object component. As we shall introduce later in Subsection 3.3.3.2, the strategy component controls how to mutate offsprings. The numbers in strategy component are initialized and generated by ES during the generations automatically, therefore they are called *endogenous* parameters. Regarding this, another *exogenous* strategy parameters are predefined before the ES generations, and treated as a constant. This is the key point of self-adaptation principle of ES, we will turn back to explain the both strategy parameters later while introducing the mutation operator. $f(\theta^{(t)})$ denotes the objective value of object component $\theta^{(t)}$, called *fitness* here. According to the "survival of the fittest" principle of the evolution theory, the fitness plays a central role in ES, which guides the direction of evolution.

The last component of an individual is K, an integer number of generations that an individual ever survived, which indicates how "old" is an individual. This parameter is related to the so-called *environment selection* mechanism, which is applied overall in evolutionary algorithms. Because at the end of each iteration, the ES select a fixed individuals to form a new population, in which the individuals in old generation are called *parent*, and newly generated individuals in current iteration are called *offspring*s. There are two well-known environment selection strategies:

1. **Comma-selection** (or *,-selection*): In comma-selection, a new population is constructed for next generation and consists of newly generated off-springs, whereas parent individuals are excluded in the new population explicitly. To increase the competition among off-springs, there are usually more off-springs generated than required, i.e. $\lambda > \mu$, where μ is fixed size of population, λ is the number of off-springs, the comma-selection is therefore characterized by $sel(\mu, \lambda)$. A bigger number of off-springs means more explorations in the search space. Nevertheless, one disadvantage of comma-selection is that the individual with best fitness ("best individual") can eventually be lost in next population, and not be found again.

2. **Plus-selection** (or $+$*-selection*, $(\mu+\lambda)$*-selection*): In plus-selection, a new population is constructed based on both parent individuals and newly generated off-springs. The plus-selection offers a competition between parents and off-springs, only the "best" ones can survive. From the optimization's point of view, the best individual would be always contained in the population. Traditionally it holds $\lambda < \mu$, so that only one small part of population will be updated, that is, parent individuals with "bad" fitness are replaced with off-springs with "good" fitness. In contrast to the comma-selection, the plus-selection concentrates on the exploitation of search space.

In ES, a generalized form of *K-selection* is proposed with a single parameter K, called *life span*. The individuals of new population can survive maximal K generations, obviously it turns out:

$$K\text{-selection} = \begin{cases} comma-selection & \text{if } K = 0 \\ plus-selection & \text{if } K = \infty \end{cases}$$

The K-selection makes a compromise between pure exploration and exploitation in search space, namely between comma-selection and plus-selection. A K-selection in ES gives of course more flexibility to control the construction of new population. Formally the environment selection of ES is defined as:

$$sel_K(P_\mu, P_\lambda) := \{\text{the top } \mu \text{ individuals in } P' \text{ according fitness}\}$$

where $P' := \{I | I \in P_\mu \text{ or } I \in P_\lambda : (\text{age of } I) \leq K\}$, P_μ denotes the current population with μ parents and P_λ a set of newly generated λ off-springs.

3.3.3.2 Mutation

One of two important operators in ES to generate a new individual is the *mutation* operator, which was originally the only operator applied in ES, as the recombination operator was introduced into ES afterward. Generally speaking, the mutation describes how to change (the component(s) of) an individual to create a new one. Based on theoretical considerations for a successful ES implementation, Beyer and Schwefel [Bey01] proposed three guiding principles:

1. *reachability* : It is the first requirement ensures that the global optimum can be reached within a finite number of mutation steps or generations from any random start solution. Only under this condition one can show the global convergence of ES;
2. *unbiasedness* : This is a requirement derived from "philosophy of Darwinian evolution". The mutation should not take use of any fitness information, but only information about the search space inherited from parents. Therefore, there is no preference among individuals, and the mutation itself should not introduce any bias;
3. *scalability* : The scalability requires the mutation step should be adaptable according to fitness and problem and progress the optimization.

A mutation operator having the above properties is proposed in [BS02], which change the object component by adding a vector with normally distributed random numbers:

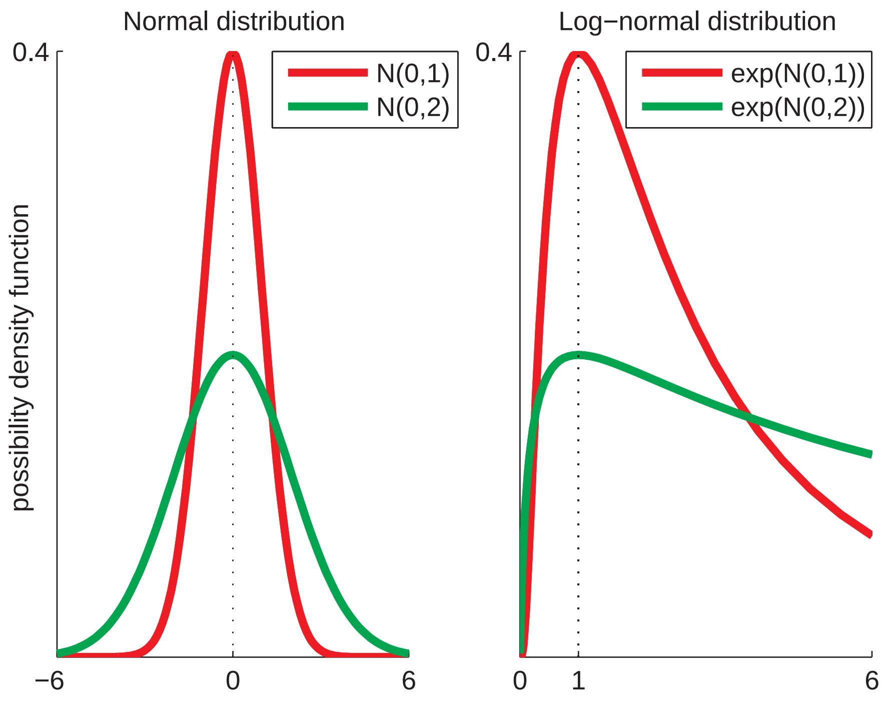
$$\theta^{(t)} \leftarrow \theta^{(t)} + [\sigma_1^{(t)} \cdot X_1, \ldots, \sigma_n^{(t)} \cdot X_n] \tag{3.3.8}$$

abbreviated as $mut_\theta(\theta^{(t)}, \sigma^{(t)})$, where $X_1, \ldots, X_n$ are randomly generated numbers following a standard normal distribution ($X_i \sim N(0,1)$, for all $i \in \{1, \ldots, n\}$), so that the modification on each parameter in an FOT follows a normal distribution of:

$$\sigma_i \cdot X_i \sim N(0, \sigma_i), \forall i \in \{1, \ldots, n\}$$

Figure 3.13 shows the probability density function of normal distributions ($N(0,1)$ and $N(0,2)$). The adaptation of any parameters involved in FOT is therefore symmetrical and has a domain $(-\infty, +\infty)$, which fulfills the requirements of reachability and unbiasedness mentioned before.

Figure 3.13: Distributions used in mutation operator



$\sigma_1, \ldots, \sigma_n$ are endogenous parameters in strategy component, they are called *step size*, since they determine the variance of the normal distribution of object components. Before applying 3.3.8, these step sizes (namely the strategy component) are modified under the principle of scalability for purpose of self-adaptation, where it is mutated by multiplying a log-normally distributed random numbers, that is:

$$\sigma^{(t)} \leftarrow e^{\tau_0 Y_0}[\sigma_1^{(t)} \cdot e^{\tau \cdot Y_1}, \ldots, \sigma_n^{(t)} \cdot e^{\tau \cdot Y_n}] \tag{3.3.9}$$

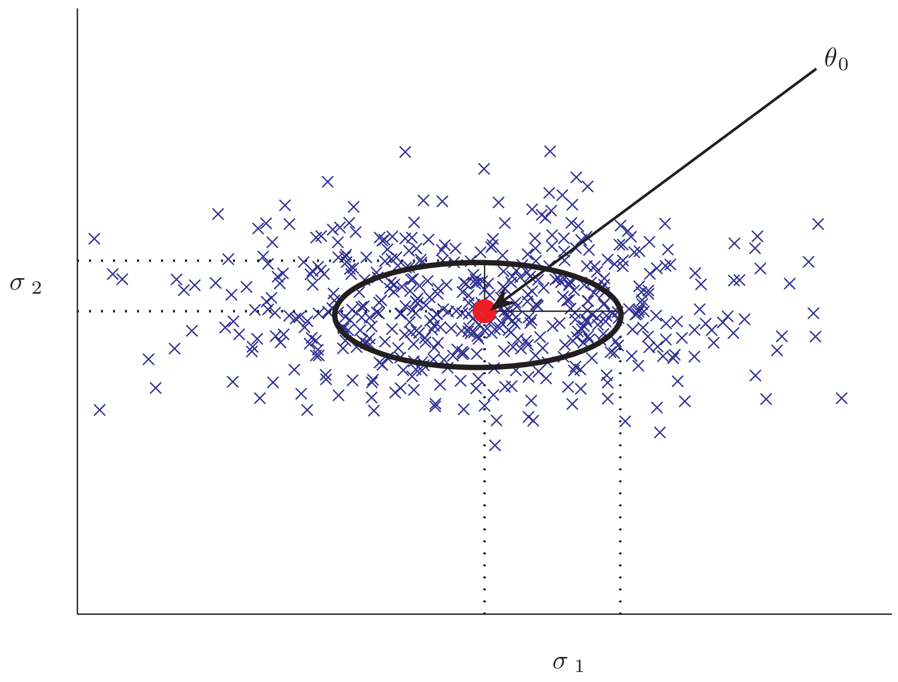
abbreviated as $mut_\sigma(\sigma^{(t)}, \tau_0, \tau)$, where Y_0 to Y_n are again randomly generated numbers following a standard normal distribution. τ_0 and τ are predefined exogenous parameters, which, as suggested in [BS02], can be selected as:

$$\tau_0 = \frac{1}{\sqrt{2n}}, \ \tau = \frac{1}{\sqrt{2\sqrt{n}}}$$

for a problem with n-dimensional object component. Note that the adaptation of strategy component by Equation 3.3.9 follows the log-normal distribution, so that ensures that all step sizes can take values in the domain $[0, +\infty)$, as illustrated in the right half in Figure 3.13.

To this end, this standard mutation operator meets all requirements suggested in [BS02]. The possible distribution of off-springs produced by mutation is demonstrated in Figure 3.14 according to two step size (σ_1 and σ_2), where an ellipse shows the expected positions of off-springs mutated on θ_0.

Figure 3.14: Distribution of off-springs by mutation



3.3.3.3 Recombination

As mutation modifies an individual to get a new off-spring with a randomly generation self-adaptation, recombination shares the information from a fixed number ρ of parent individuals. Unlike the standard crossover operator in GA, where two parents produce exactly two off-springs, the standard recombination in ES generates only one offspring upon a set of parents, see Figure 3.15, where two standard classes of recombination used in ES are illustrated, namely:

1. **Intermediate recombination** : Formally given a set of ρ parents $(I_1, I_2, \ldots, I_\rho)$,

a new offspring produced by recombination, I_{rec}, is defined:

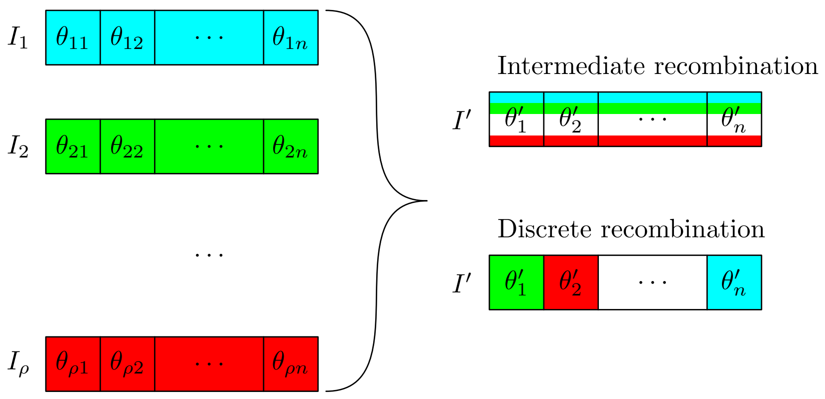
$$\begin{aligned} I_{rec} &= (\theta_{rec,1}, \theta_{rec,2}, \ldots, \theta_{rec,n}, \sigma_{rec,1}, \sigma_{rec,2}, \ldots, \sigma_{rec,n}, \text{ Null}, 0) \\ \text{where } \theta_{rec,k} &= \frac{1}{\rho} \sum_{i=1}^{\rho} \theta_{i,k} \text{ and} \\ \sigma_{rec,k} &= \frac{1}{\rho} \sum_{i=1}^{\rho} \sigma_{i,k}, \text{ for all } k \in \{1, \ldots, n\} \end{aligned}$$

where Null means the newly generated offspring is still not evaluated, and 0 indicates its age. From the geometric point of view, I_{rec} is the gravity center of ρ parents. In case of discrete object component instead of numeric one, an additional procedure is needed to map back newly generated component into discrete domain, such as rounding.

2. **Discrete recombination** (also referred as *dominant recombination*) : In contrast to intermediate recombination, the discrete recombination takes one parent into account each time to determine one of the object or strategy component, where the parent is selected randomly under a uniform distribution, that is:

$$\begin{aligned} I_{rec} &= (\theta_{rec,1}, \theta_{rec,2}, \ldots, \theta_{rec,n}, \sigma_{rec,1}, \sigma_{rec,2}, \ldots, \sigma_{rec,n}, \text{ Null}, 0) \\ \text{where } \theta_{rec,k} &\in \{\theta_{1,k}, \theta_{2,k}, \ldots, \theta_{n,k}\} \text{ and} \\ \sigma_{rec,k} &\in \{\sigma_{1,k}, \sigma_{2,k}, \ldots, \sigma_{n,k}\}, \text{ for all } k \in \{1, \ldots, n\} \end{aligned}$$

Figure 3.15: Recombination in ES


I_1
θ_{11} θ_{12} $\cdots$ θ_{1n}
I_2
θ_{21} θ_{22} $\cdots$ θ_{2n}
$\cdots$
I_ρ
$\theta_{\rho 1}$ $\theta_{\rho 2}$ $\cdots$ $\theta_{\rho n}$
Intermediate recombination
I'
θ'_1 θ'_2 $\cdots$ θ'_n
Discrete recombination
I'
θ'_1 θ'_2 $\cdots$ θ'_n

For both recombination variants, the recombination operator can be generalized as $rec(P_\rho)$, where P_ρ are randomly selected ρ parents in current population P. Differ from the environment selection for purpose of constructing a new population for next generation, the selection of parents for recombination operator is unbiased, in which all parents in current population are selected with equal probability (again under a uniform distribution), independent of their fitness.

3.3.3.4 Termination

To terminate the ES loop, different stop criteria can be integrated into an ES, some commonly used termination criteria are follows:

1. *maximal generation* : To limit the number of generations, this criterion gives no guarantee that an optimum can be found by ES, but it is useful to limit resources available to ES, and can be applied, for example, in distributed computing.
2. *maximal runtime* : Just like the previous one, the maximal runtime is used to limit the maximal running time for an ES.

3. *expected fitness* : Since in practice one interests usually in searching a solution with “adequate” fitness, rather than an absolute global optimum, an expected fitness here serves exactly for this purpose.

4. *maximal stalling* : In order to approximately detect whether an ES converges into an optimum, the maximal stalling gives some measurements for this purpose. For example, if in last T generations there is no improvement achieved, namely no individual with better fitness is found, then an ES is treated as converged into an optimum, or in the case the different between “best” and “worst” individuals in last generation falls below a user-defined threshold, and so on.

3.3.3.5 Evolution Strategies Algorithm

Now we can give the complete ES algorithm in pseudo-code, shown in Algorithm 3.3. In line 8 the recombination operator is called to generate a new offspring, whose object and strategy components are marked as $\theta_i^{'(t)}$ and $\sigma_i^{'(t)}$ for further operations. In line 9 and 10 these both components are mutated using the standard mutation operator, and their relative positions of both mutation operators (first mut_σ, then mut_θ) play an important role to meet the reachability and unbiasedness, as mentioned before. As the last two parameters in the ES, namely τ_0 and τ, can be approximated according to the dimension of object component, the ES algorithm is usually referred as (μ, λ, K, ρ)-Algorithm with its four characteristic parameters.

Algorithm 3.3: Evolution strategies algorithm

```
Input: Population size $\mu$, number of off-springs $\lambda$, life span $K$, number of
       parents selection $\rho$, exogenous parameters $\tau_0$ and $\tau$
Output: Solution $p$
1  begin
2      $t \leftarrow 0$ ;
       // initialize
3      $P^{(t)} \leftarrow \{(\theta_i^{(t)}, \sigma_i^{(t)}, f(\theta_i^{(t)}), 1) | i = 1, \ldots, \mu\}$;
4      while (Not terminated) do
5          $P_{off} \leftarrow \emptyset$ ;
6          for ($i \in \{1, \ldots, \lambda\}$) do
7              $P_\rho \leftarrow$ select $\rho$ parents from $P^{(t)}$ ;
8              $[\theta_i^{'(t)}, \sigma_i^{'(t)}] \leftarrow rec(P_\rho)$ ;
9              $\sigma_i^{''(t)} \leftarrow mut_\sigma(\sigma_i^{'(t)}, \tau_0, \tau)$ ;
10             $\theta_i^{''(t)} \leftarrow mut_\theta(\theta_i^{'(t)}, \sigma_i^{''(t)})$ ;
11             evaluate $\theta_i^{''(t)}$ to get $f(\theta_i^{''(t)})$ ;
12             generate an offspring $(\theta_i^{''(t)}, \sigma_i^{''(t)}, f(\theta_i^{''(t)}), 0)$ into $P_{off}$ ;
13         end
           // Construct a new population
14         $P^{(t+1)} \leftarrow sel_K(P^{(t)}, P_{off})$ ;
15         increase age of individuals in $P^{(t+1)}$ by 1;
16         $t \leftarrow t + 1$;
17     end
18     $p \leftarrow$ the object component of best individual in $P^{(t)}$ ;
19 end
20 return $p$
```

Note that the parameters involved in an FOT have generally different domains, for example, the parameter of a Hamacher t-norm is only valid in the interval $[0,+\infty]$. However, the parameters of offsprings might lie out of corresponding domains due to the recombination and mutation operators. To avoid generating offsprings with invalid parameters, a validation step is integrated into the recombination and mutation operators implicitly, which validates the parameters of newly generated offsprings and adjusts the parameter values if necessary. For instance, if a Hamacher t-norm is assigned to a negative parameter, the validation step would correct it into its definition domain.

3.4 Experimental Evaluation

To evaluate the performance of the calibration techniques discussed in this chapter, a number of experiments are arranged in this section. We conduct several experimental studies to investigate the performance of calibration techniques of FOTs. All experiments are carried on a computer with following configurations: Intel Dual-core 2.40GHz CPU; 2GB RAM; Microsoft Xp operating system.

To test the calibration techniques, two especially important questions are addressed here:

1. Efficiency : Are these methods able to optimize FOTs within acceptable requirements, such as runtime? Of course a fair comparison among these techniques is desired to help applying FOT in practice. Beside this, the performance of FOTs by handling discrete data (input/output) shall be investigated, which is often used in many applications. To make these possible, we make assumption that the training data are actually generated by FOT models, the calibration techniques are set to try to find these hidden FOTs with given training data.
2. secondly we want to compare FOTs with other model classes in a learning (classification) context, in order to show the advantages by applying FOTs in case where expert prior knowledge about the qualitative structure of an evaluation scheme is available.

3.4.1 Synthetic Data

We shall give the details how to generate synthetic data and related terminologies at first, then a comparison among mentioned calibration techniques is conducted.

3.4.1.1 Data Generation

In this section, the synthetic data is generated by FOTs, we design the calibration problems by varying the component complexity of FOTs, that is: Given a fixed complexity in terms of a number of interior nodes, say C, an FOT is generated by starting from a root node, and iteratively adding leaf nodes until the FOT has exact C interior nodes. Algorithm 3.4 shows this procedure, where in line 2 and 6 an aggregation operator is selected randomly under the uniform distribution, namely each kind of aggregation operator (Hamacher t-norm, t-conorm and mixnorm) has $\frac{1}{3}$ probability to be selected, then its parameters are initialized randomly in corresponding domains. In case of OWA mixnorm, we have to ensure that the sum of parameters must be one, for this purpose, there are firstly $n-1$ random numbers generated in $[0,1]$, these sorted $n-1$ points can divide the unit interval into n parts, then the length of each part is assigned to each parameter in an OWA mixnorm; in case of a Hamacher t-norm or t-conorm, the parameter must be a positive number (domain: $[0,+\infty)$). For convenience of use, this parameter is generated randomly in a fixed domain $[0,10]$ by default. In line 5, the position for newly generated aggregation operator is selected randomly, again it follows a uniform distribution, where all possible positions can be selected with equal probability. Visually Figure 3.16 shows an example for this selection, where A denotes an aggregation operator, and dashed circles represent all possible positions. For the sake of simplicity, we restrict aggregation operators have exactly two children in this section, so that FOTs can be viewed as binary trees, the number of children of aggregation operators is of course extendable in practice. In line 10 fuzzy sets are inserted at positions of leaf nodes to complete the construction of an FOT, we apply in this section the trapezoid membership function as standard case to define fuzzy sets. Since a trapezoid membership function is characterized by four parameters, we generate four parameters randomly following a uniform distribution in a predefined domain (default: $[0,100]$), then they are sorted ascendingly, and the two parameters in the middle build the core of trapezoid membership function, whereas the smallest and biggest parameters build the support of trapezoid membership function.

Algorithm 3.4: FOT generator

```
    Input: Complexity $C$
    Output: An FOT $F$
 1  begin
 2      $F \leftarrow$ createNode() ;
 3      $c \leftarrow 1$ ;
 4      for $c < C$ do
 5          $p \leftarrow$ getRandomPos($F$) ;
 6          $a \leftarrow$ createNode() ;
 7          insert $a$ into $F$ at position $p$ ;
 8          $c \leftarrow c+1$ ;
 9      end
10      fill $F$ with fuzzy sets.
11  end
12  return $F$
```

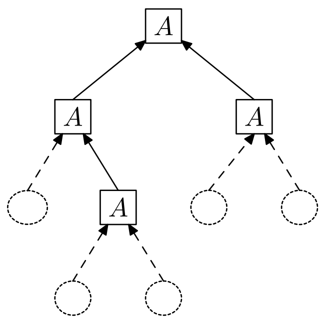


Figure 3.16: Selection a random position for FOT generator

After the tree structure of an FOT has been determined, a training data can be produced by generating a set of random input vectors, and using FOT to compute the corresponding outputs, as the pseudo-code in Algorithm 3.5 shows.

Since the synthetic data in this section is generated from FOT models, there does exist a parameter specification, so that a corresponding FOT makes no error on the synthetic data. In following text, we check the aforementioned calibration techniques whether they are able to find this parameter specifications of FOTs.

Algorithm 3.5: Training data generator

```
   Input: Complexity $C$, training data size $N$
   Output: Training data $D$
 1 begin
 2     $F \leftarrow$ random generated FOT with $C$;
 3     $D \leftarrow \emptyset$ ;
 4     $i \leftarrow 0$ ;
 5     for $(i \leftarrow N)$ do
 6         $x \leftarrow$ random generated input vector for $F$;
 7         $D \leftarrow D \cup [x, F(x)]$ ;
 8         $i \leftarrow i + 1$ ;
 9     end
10 end
11 return $D$
```

3.4.1.2 Comparison of Calibration Techniques

Our first experiment aims to gain an impression on performance of four studied calibration techniques ($GD, GD-direct, SA$ and ES), in which for a randomly generated FOT, say $F(x,\theta)$, a candidate solution represented by a set of parameter in F is initialized randomly for these techniques, then a training data based on $F(x,\theta)$ is generated and delivered into these techniques, the performance of these techniques is evaluated from following aspects:

- **Mean squared error** (*MSE* for short) : As the objective function of the calibration of FOTs is introduced, we mention that squared error is the default used loss function. Assume that the output of a calibration technique is $F(x,\hat{\theta})$, the mean squared error of $F(x,\hat{\theta})$ is defined as:

$$MSE := \frac{1}{N} \sum_{(x,y)\in D} (F(x,\hat{\theta}) - y)^2$$

  where N is the size of training data D, $\hat{\theta}$ is the parameter specification of F. Notice that *MSE* gives greater weight to bigger difference between the real and estimated outputs than smaller one.

- **Success rate** (*Suc* for short) : Since the parameter domain we considered in calibration of FOTs is a continuous one, we use success rate to indicate whether a solution lies near a global optimum closely enough. A calibration is treated as successful, if the achieved *MSE* of its output is small enough, that is, its $MSE \leq \varepsilon$, where ε is a threshold, $\varepsilon = 0.01$ as default. Formally:

$$\text{Suc} := \begin{cases} 1 & \text{if } MSE \leq \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

- **Runtime** (*Time* for short) : To compare the efficiency of mentioned calibration methods, the running time of calibration process is recorded here.

Since the performance of calibration is strongly dependent on the randomly generated initial solution(s), we use a random *restart* technique here, in which the whole calibration process can be restarted until to a fixed time, as long as it is not succeed, namely failed to find a model with small enough *MSE*. Finally the best among all outputs from each restarted calibration processes is adopted. For gradient descent and simulated annealing methods, a relative large restart number is allowed (default 10), while the small restart number for *ES* is set (default 3), because the influence of randomness is smaller for the *ES* than other variants, where we maintain a population that is initialized randomly.

In this experiment, all results are obtained based on 50 randomly generated FOTs with 200 training examples, as the complexity of FOTs varies from 1 to 60. The configurations for these calibration methods are listed in Table 3.2, and the comparison results are shown in Figure 3.17. As one can see, the *ES* outperforms the other three variants, which shows a reliable high success rate, although the success rate decreases a little as complexity becomes large. Neither gradient descent nor simulated annealing can get close to a global optimum as complexity becomes large, obviously because they get stuck in local optima. One interesting observation here is that the $GD-direct$ provides better results comparing to the *GD*, since the *GD* propagates the overall error backward by estimating errors for interior nodes, this estimation brings in turn error in the calibration. Unlike in the $GD-direct$ method, using the chain rule to compute partial derivatives is error-free from the mathematical point of view, all parameters are updated based on the overall error simultaneously, which decreases the error on training data. Of course the training time of *ES* is not comparable with other variants, but still in an acceptable range (maximal $62.3s$).

Table 3.2: Configurations for calibration methods

Name	**Value**	**Remark**
For gradient descent (*GD* and $GD-direct$)		
batch learning	*Yes*	
type	0	direct gradient descent
For simulated annealing (*SA*)		
learning rate	0.1	decrease during iteration
temperature	$\frac{1}{t}$	t the number of iteration
For evolution strategies (*ES*)		
population size μ	10	
#off-springs λ	20	
#parents for recombination ρ	2	
life span K	100	
recombination type *rec*	intermediate	
max generation	100	
stall generation	10	

It is not surprised that the both greedy variants *GD* and $GD-direct$ fail in calibration of FOTs, since they often trap in the local optimum despite applying the restart technique, which tries to avoid local optimum by using different initializations. We have arranged more experiments under various parameter settings on *GD* and $GD-direct$, for example turning to online learning in stead of batch learning, changing the type of update rules, and so on, there is no significant improvement achieved comparing with the standard settings. Therefore, in following discussion we shall ignore these two variants.

On the other hand, the *SA* also fails in this case, which is well-known for its ability to escape from local optima and finally might reach the global optimum [LA87]. This might due to two reasons: Firstly we notice that the problem of calibration of FOTs has extreme many local optima as the complexity of FOT becomes large, which can be observed in that *GD* often fails in finding a global optimum despite various initializations; Secondly the parameter settings of *SA* in this case are selected blindly, which is of course not sufficient for success of *SA*. Although it is possible to tune

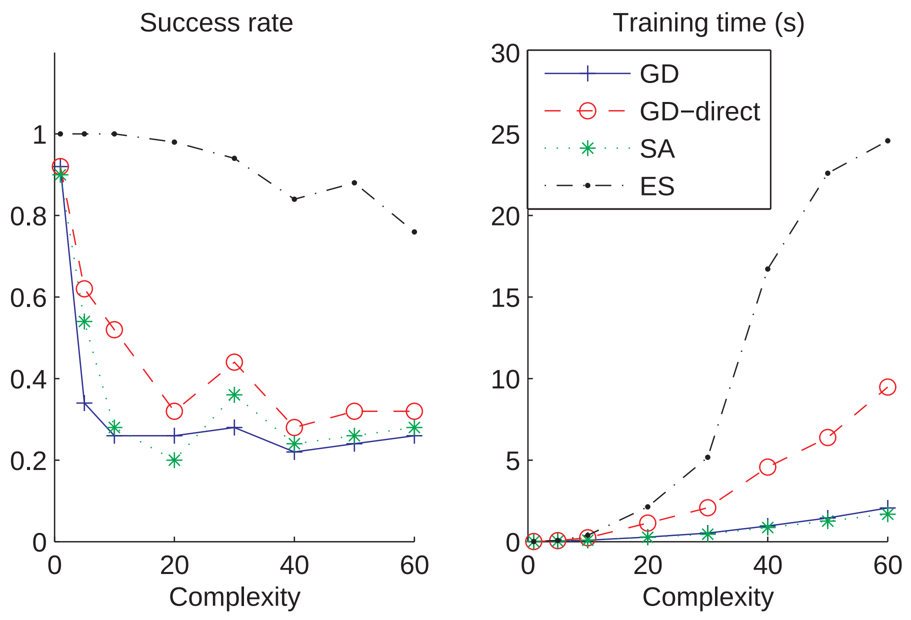


Figure 3.17: Comparison result on calibration techniques

the parameter settings of *SA* to increase its success rate, but it might also increase the runtime of *SA* at the same time [Haj88]. From another aspect, *SA* and single trial versions of *ES* ($(1,1)-ES$, namely with population size 1, comma selection and offspring size 1) have a close relationship on optimization over continuous variables, as Rudolph indicated in [Rud93], which only differ in the choice of the sampling and acceptance distributions, namely how to generate a new solution and which rule to accept it. Following the observations in [Rud93], the different between *SA* and *ES* becomes less essential by designing massively parallel *SA* algorithm. Therefore we shall concentrate on *ES* in following experiments and ignore *SA* too.

3.4.2 Parameter Determination in Evolution Strategies

Applying *ES* is usually computationally expensive, especially as applied to optimize real-world problems, so the optimization on parameters in *ES* is always advisable, this issue has been addressed in several scientific disciplines in last decades in terms of *experimental design* [CMG$^+$93, Joh02, Kle01, Mor02, BB03, BB05] or *ES design*

The experiments in this section aim to find out how the main parameters involved in *ES* influence the overall performance. It is clear that no universal "optimal" parameter specification exists for the calibration of FOTs for any case, however we want to gain a general impression on important parameters in *ES* through these experiments. In Table 3.3 the primary parameters of *ES* investigated in this section are listed, as well as the respective domains and default values suggested in [Bäc96].

Table 3.3: Primary parameters in ES

Name	**Remark**	**Domain**	**Default**
μ	the population size	$\mathbb{N}_+$	15
λ	the number of off-springs	$\mathbb{N}_+$	105
ρ	number of parents for recombination	$\{1,\ldots,\mu\}$	15
rec	recombination type	{discrete, intermediate}	intermediate
K	life span	$\mathbb{N}_+$	1

The goal of *ES* design is to find a parameter specification (also called a *design point* in this context) for these five parameters, say $\phi \in \Phi$ with Φ denotes the domain of parameters, so that an *ES* specified with ϕ can optimize FOTs on given training data as well as possible. Note that a training data must be given and seen as fixed for *ES* design, so that the learnt parameter specification is only optimal for the given training data. The goodness of a parameter specification can be measured with different criteria over the training data, for instance, *MSE* is used to get the objective value of a parameter specification. The experimental results of *ES* may reveal information about the robustness of *ES*, which lead to new insights in the role of parameters considered here, for example, the relationship between μ and λ, etc.

Since there are infinite potential parameter specifications in Φ, it is impossible to try a complete enumeration of all parameter specifications. Instead, let us consider several commonly used methods for this purpose:

- **One-parameter-at-a-time** : This method tries to vary one certain parameter at a time, while the remaining parameters are held constant. It provides an estimate of the influence of a single parameter at selected fixed conditions of the other parameters. To run this method, one just simply need to decide the order of parameters to be varied, and how to optimize single parameter while others with constant values. However, it only works under the assumption that the effect would be the same at other specifications of remaining parameters, which requires in turn that effects of parameters behave additively in terms of the objective value in respective domain. Furthermore it is difficult to determine interrelationship between parameters. So we do not recommend to use this method.

- **Statistical design of experiment:** (DOE) This method has a long tradition in statistics and has been developed for different applications in several areas, such as computer simulation and deterministic computer experiments [Kle01, BB05]. Let us review the basic idea of DOE at first. Generally if we consider the relationship between design points and their objective values (default *MSE* here), then a mathematical model can be represented as follows:

  $$y = f(\phi)$$

  where $\phi \in \Phi$ is a design point and y is its objective value (*MSE*) respectively, f is a mathematical function, e.g. $f : \mathbb{R}^q \to \mathbb{R}$ for a design point with q real-valued variables and the objective value of each design point is again a real number. In order to find an optimal design point, we try to estimate the mathematical function f at first, then find an optimal design point based on f. Since the objective value of a design point is real-valued here, estimating f becomes a typical regression analysis problem [HMS02]. Therefore, we try to approximate f by using different regression models, such as linear or quadratic models.

  Let us demonstrate how to use DOE to design an experiment with q parameters. Independent of the type of regression model applied in DOE, several design points (called *samples*) are selected and their objective value are calculated, the relevant question here is how many samples are necessary and how to select them. Assume that there are n samples are selected, then the objective value and design point for i-th sample can be expressed as y_i and ϕ^i, for all $i \in \{1,\dots,n\}$. DOE applies simply a linear regression model to approximate f, which is represented in following form:

  $$y = X \cdot \beta + \varepsilon$$

  where $y = [y_1,\dots,y_n]'$ and β is the vector with $1+q$ coefficients, ε denotes the vector of n error terms, X stands for the $(n \times (1+q))$ matrix of independent regression variables, which is constructed as:

$$X := \begin{pmatrix} 1 & \theta_1^1 & \dots & \theta_q^1 \\ 1 & \theta_1^2 & \dots & \theta_q^2 \\ \vdots & & & \vdots \\ 1 & \theta_1^n & \dots & \theta_q^n \end{pmatrix}$$

where θ_i^j is the i-th parameter in j-th sample. Using least square methods to estimate this linear model, one follows the optimal regression coefficient as:

$$\hat{\beta} = (X'X)^{-1}X'y$$

with X' is the matrix transpose of X, and X^{-1} denotes the matrix inverse of X. To simplify the computation of $\hat{\beta}$, each variable would be scaled to the range $[-1,1]$, called *standardized* here, then only the two endpoints of each variables are considered for constructing design points, which results that X consists only of -1 or 1. Furthermore, one of typical designs in DOE, *factorial design*, takes 2^q samples by combining available endpoints of each variable, so that X is an orthogonal matrix, namely $(XX') = nI$ with I is the identity matrix of size n. Beginning at the middle point $\theta = [0,0,\dots,0]'$, to get a design point with smaller objective value, a linear search is performed in the direction of the steepest descent, which is given by $-(\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_q)$, note that the component with index 0 of $\hat{\beta}$ is ignored here, since it is a constant coefficient. To regulate the step sizes $\Delta\theta_i$, the largest absolute regression coefficient is selected: $j = \arg_i \max(\hat{\theta}_i)$, then the update for other parameters can be scaled as:

$$\Delta\theta_j = -\frac{\hat{\theta}_i}{|\hat{\theta}_j|/\Delta\theta_i}$$

for all $i \in \{1,\dots,q\}$. Along this direction, further design points can be examined, until a local optimum is found (further update decreases the objective value or exceeds the boundaries).

Figure 3.18 demonstrates the basic idea of DOE for optimization problem based on two standardized parameters θ_1 and θ_2. Four samples are taken in this case at the corners of boundaries, from which the regression lines are estimated (denoted with dashed lines). From the start point d_0, several new design points along the direction of steepest descent are selected and evaluated. The result of DOE, say d_0', can be improved further by setting d_0' as central point and constructing another DOE process.

Any parameter involve in DOE has to be standardized into the interval $[-1,1]$, this is quite easy for quantitative parameter, for example the population size, say μ, ranges in $[l,h]$, then it can be standardized as μ', using the linear transformation $\mu = (l+h)/2 - (l-h)\cdot\mu'/2$. Qualitative parameters, such as recombination operator, have to be treated separately, because a linear search cannot be performed in this case.

The factorial design of DOE requires at least 2^q samples, which becomes computational expensive as the number of parameters to be optimized becomes large. Another *fractional factorial designs* requires only 2^{q-k} samples by neglecting interactions between some (k) parameters [Kle01]. Other designs have also been proposed, such as *central composite designs* by adding central points and axial points. It has been shown that DOE is more efficient than one-parameter-at-a-time strategy [Kle87].

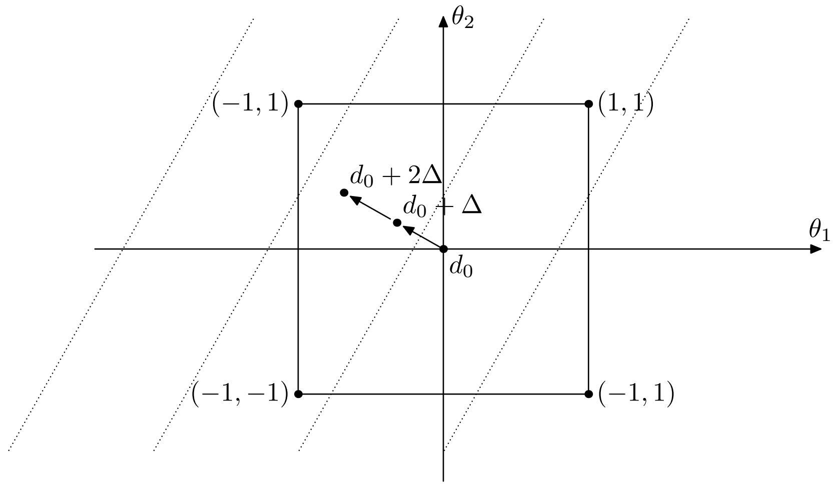


Figure 3.18: Path of Steepest Ascent in DOE

- **Design and analysis of computer experiments** (DACE) : In order to discover the nonlinear dependencies, the DACE [LNS02, SWN03] assumes that the correlation between errors is related to the distance between the sampling points, whereas regression models used in DOE assume that the errors are independent. Beside this, DACE can use different regression models to approximate the mathematical function f, for example, quadratic or linear models.

  The DACE considers samples placed in the interior of the design space, rather than only on the boundaries as the DOE does. To determine N sampling points for DACE, the *Latin Hypercube Sampling* (LHS) method [MCB79] is selected, which provides a great flexibility in choosing the number of samples. In LHS the interval of each parameter is partitioned into N sub-intervals with identical lengths, so that there are exactly N^q rectangles for q parameters. Then N rectangles are selected randomly following the uniform distribution, and one design point is generated in each selected rectangle again following the uniform distribution. Generating design points in this way is called *Latin Hypercube Design* (LHD).

  However the problems arise from DACE designs by dealing with qualitative factors, even with the LHD, as Santner argued in [SWN03]. Moreover, the advantages and disadvantages of using DACE are discussed in [JSW98] elaborately.

- **Sequential parameter optimization** (SPO) : Another method for the purpose of experiment design is to generate the design points not at once, but sequentially, which might profit from that the selection process for further design points can include knowledge from the evaluation of previously generated design points. Since several works [SWN03, BB05] have shown that SPO outperforms other alternatives, especially for evolution strategies, we shall apply SPO to determine the parameters of *ES* in this section.

  Sequential sampling approaches with adaptation have been proposed for DACE [SWN03, SWMW89], here let us review the basic idea of Thomas Bartz-Beielstein [BBLP05], which is also used in this section to determine the parameters of *ES*. As the name indicates, SPO begins with several design points selected by using LHS, the evaluation results on these design points are used to generate first DACE model. Based on this first DACE model, further design points are selected, and the DACE model would be improved with newly evaluated design points. The crucial issue of this method is how to decide next design point based on a DACE model. In Santner et al. [SWN03], a heuristic algorithm for unconstrained global minimization problems is presented. Let us denote y^k_{min} the smallest known objective value after k design points, for a new design point ϕ, the improvement of ϕ is defined as:

$$I(\phi) = \begin{cases} y^k_{min} - y(\phi), & \text{if } y^k_{min} > y(\phi) \\ 0 & \text{otherwise} \end{cases}$$

with $y(\phi)$ the objective value of ϕ. As the exact value of $y(\phi)$ is unknown, the goal is to optimize its expected value, the so called *expected improvement*, an alternative to estimate expected improvement has been proposed in [Sch97]:

$$E(I(\phi)) = \int_{-\infty}^{y^k_{min}} (y^k_{min} - f(\phi)) \cdot \psi(f(\phi)) d\phi$$

with $\psi(\cdot)$ as a probability density function representing uncertainty about $f(\phi)$. Using DACE one can determine the mean squared error and estimation of $f(\phi)$ for a newly selected design point ϕ, which enable to compute $E(I(\phi))$ without solving the integral. Regarding the values of $E(I(\phi))$, a new design point ϕ' is selected in a design space, which is expected to contain special “good” design points or have high uncertainty. The newly selected design point would be evaluated afterward, and inserted into DACE model. In this way, a DACE model would be improved iteratively, until a design point with adequate quality is found or a maximal number of iterations is reached.

The experiments to determine parameters of *ES* in this section are based on a *sequential parameter optimization toolbox* (SPOT), this is a MATLAB statistics toolbox developed over the last years by Thomas Bartz-Beielstein et al[1]. Adapting SPOT settings to determine parameter specifications of *ES* is very simple, where one just need to point out the parameters to be determined and specify their domains. There are two configuration files for SPOT:

1. the *ROI* file, which gives the parameters to be determined in SPOT and their domains. Regarding the goal of this section to determine five primary parameters of *ES*, a *testRobq.roi* file is designed with following content:

   ```
   name low high isint pretty
   p 2 60 TRUE 'population size'
   parents 1 20 TRUE 'number of parents'
   lifeSpan 0 500 TRUE 'lifespan'
   lambda 2 120 TRUE 'off-springs'
   ```

   A *roi* file has a clear and comprehensive syntax, so that the previous file can be easily understood, e.g. the first parameter to be determined is the *population size*, in which the interest range lies in [2, 60], etc. Here the interest ranges for four quantitative parameters are specified, in which the default setting suggested in [Bäc96] is already contained. Note that the recombination type, the sole qualitative parameter, is treated separately, since linear regression model is not proper to approximate qualitative parameters. In this case, we simply compare the performances under two alternative recombination type (*discrete* and *intermediate* recombination), and choose the better one regarding their performances.

[1] Free available under http://www.gm.fh-koeln.de/ bartz/experimentalresearch/spot03.zip

2. the configuration file for SPOT, which has an extension name like other MATLAB scripts, e.g. *testRobq.m* in this case. The parameters to configure SPOT are given in this file, e.g. the number of repetitions, or the parameter ρ for SPO as mentioned before.

Since the previous works in this thesis are implemented in Java, to evaluate a design point selected by SPOT, a small modification should be made in the file *demoSpotJava.m*, a template file designed specially for Java application, where the evaluation process is called through:

```
dos('java -cp robq.jar robq.experiment.ExperimentSPOT');
```

To guarantee a fair comparison between different design points, special considerations are taken on the stop-criteria of *ES*. Some stop-criteria are not appropriate for this task, for instance, the maximal number of generation, apparently a design point with larger population and offspring sizes has more chance to get a smaller objective value than one with smaller population and offspring sizes, since the former one can simply try more individuals than the later one. From this aspect, only "fair" stop-criteria are allowed to evaluate design points from SPOT, such as the maximal runtime of *ES*. Moreover, a new stop-criterion is defined to limit the maximal times to compute fitness of individuals, called *maximal evaluation time*. Since the fitness of any individual is calculated based on given training data, this is the most time-consuming part of *ES*, as the mount of provided training data becomes large. The maximal evaluation time gives an upper bound for scanning over training data.

As arranging experiments in SPOT, the results can be given in different form. In graphical form, the interrelationship between any two parameters can be shown in a 3-dimensional space regarding the objective value. As example, the SPOT is applied to determine parameter specification of FOTs with fixed complexity (50 here), that is, an FOT is generated randomly with 50 aggregation operators and a synthetic training data is then created respectively (with 200 instances), SPOT is applied to determine the four quantitative parameters of *ES* (the recombination type is set to *discrete*). Figure 3.19 shows the distribution of objective values based on *population size* and *parents*, where the black points represent evaluated design points. It is clear to see that "better" design points lie in the region with higher *population* and *parents* size, which is also plausible. Additionally, the mean square error of regression models used in SPOT can be observed, as Figure 3.20 shows.

The detailed results can be found in a *.bst* file, in which the "best" design points found so far are listed, for example a typical result file looks like:

```
Y p parents lifeSpan lambda CONFIG
0.004 50.7797 9.66162 190.904 38.0513 11
0.0049 50.7797 9.66162 190.904 38.0513 11
```

In this section, we arrange experiments according the complexity of FOTs (ranges from 10 to 60) and the recombination type. Since SPOT can only give "best" design point for a fixed problem, the experiment would be repeated up to a fixed time (10) to get relative reliable results under each configuration. To this end, the mean values of best parameters reported by SPOT are plotted for four quantitative parameters, as well as their standard deviations in Figure 3.21 and 3.22.

It is clear to see that the dependency on complexity for these parameters is not so strong in both cases, so that we do not need to extract a parameter specification for each complexity, but only take one general optimal parameter specification for FOTs

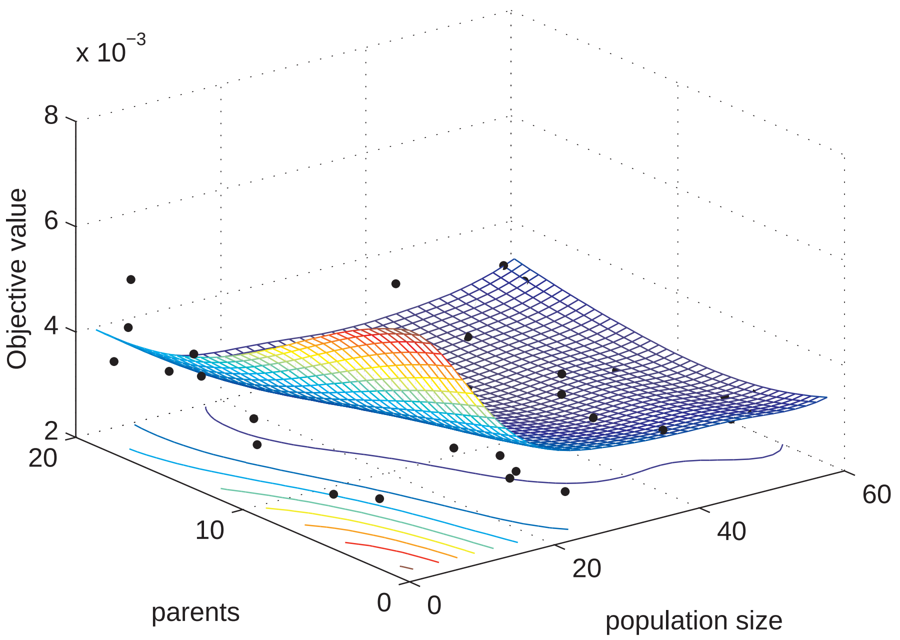


Figure 3.19: Objective values depending on two parameters (μ and ρ)

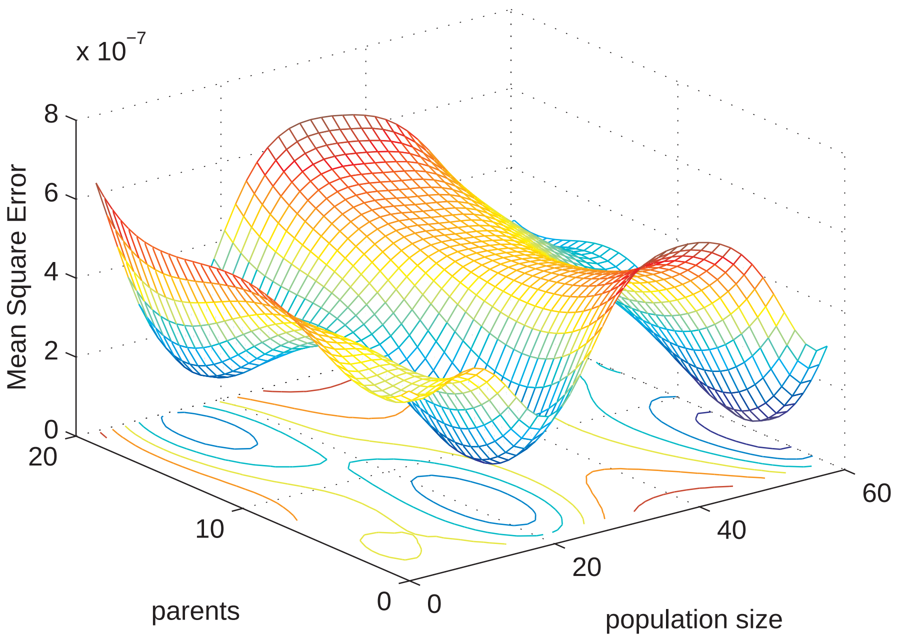


Figure 3.20: Mean square error depending on two parameters (μ and ρ)

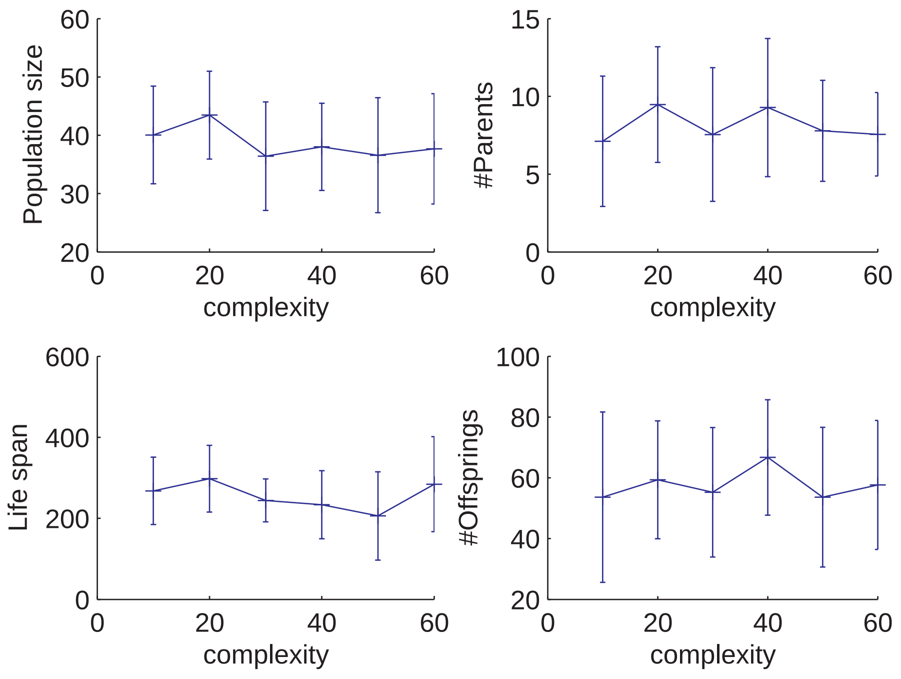


Figure 3.21: Best parameters reported by SPOT (recombination type: discrete)

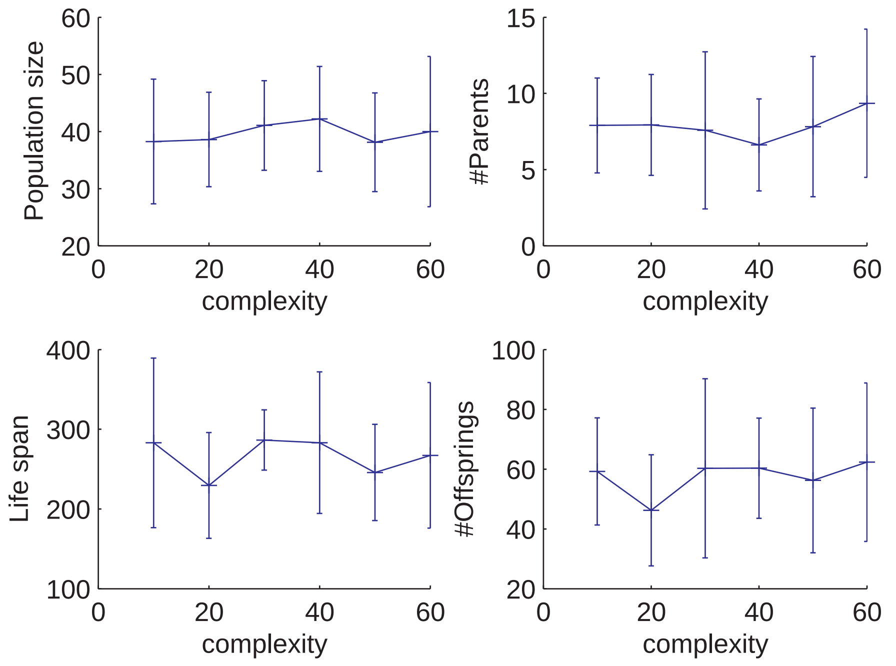


Figure 3.22: Best parameters reported by SPOT (recombination type: intermediate)

with different complexities. Considering the recombination type, the averaged objective value in case of discrete recombination (mean 0.0089, standard deviation 0.0069) is clear lower than in case of intermediate recombination (mean 0.0104, standard deviation 0.0074). We get "best" parameter settings of *ES* in general case by taking the averaged values (rounded to the nearest integer numbers) among all considered complexities, which are listed in Table 3.4. Let us refer these parameter settings as *optimized design* reported by SPOT, in order to show the improvement of optimized design, we arrange experiments to compare the optimized design and the original parameter setting listed in Table 3.2, denoted as *original design*, under the same condition just as the previous section does. The results are shown in Figure 3.23, where one can see that the performance of optimized design is clearly stabler than the original design, even as the complexity increases, while the training time is a little higher than the original design, but still in an acceptable range (around 1 minute for an FOT with 60 interior nodes). In following experiments, *ES* is specified with parameters listed in Table 3.4 without further remarks.

Table 3.4: Best parameters recommended by SPOT for ES

Name	**Value**
population size μ	40
#off-springs λ	60
#parents for recombination ρ	8
life span K	255
recombination type *rec*	discrete

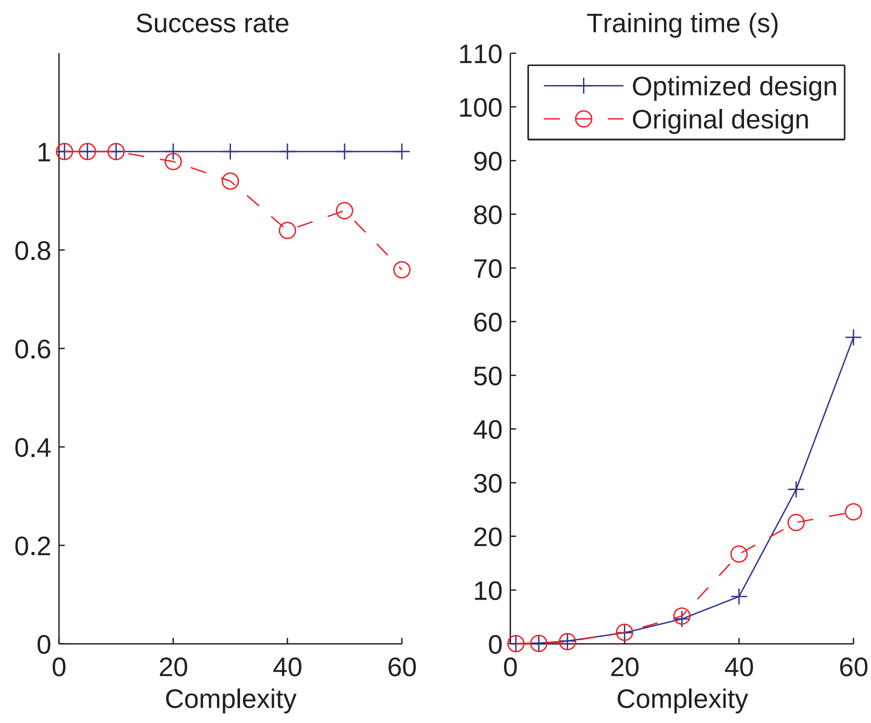


Figure 3.23: Comparison between original design and optimized design recommended by SPOT

3.4.3 Noisy Data

With synthetic data we have simulated an "ideal" case for the calibration problem of FOTs, namely the prior knowledge of experts can be modeled in the form of FOT perfectly, and training data provided by experts are also precise, in the sense that a special FOT exists, which mimic the behavior of experts by assessing training data correctly. However the condition for such "ideal" case with "precise" training data can hardly be observed in practice, for example, human experts prefer to give categorical rather than numeric assessment, or the uncertainty over quality might lead to error in training data, etc. In this experiment, we consider error in training data, in which we add random noise into the training data to test the calibration process on noisy data. In the next section, we shall examine the ability of the calibration process to handle categorical data.

The noise is simulated in this case as a random variable, say e, which follows normal distribution with mean 0 and a positive standard deviation σ, namely $e \sim N(0,\sigma)$. This noise is added into training data to deflect the real output, that is, the quality of an input vector x in this case, say y, is set as:

$$y = \max(\min(F(x)+e,1),0)$$

with $F(x)$ denotes the real output by inputting x into an FOT F, and e a randomly generated error. Due to the embedded noise in training data, the measurement used to compute success rate becomes too strict. It is necessary to relax the condition in this case, which can be done by assigning a bigger ε, or taking the embedded noise into account, this leads to a modified success rate, defined as:

$$\text{Suc} = \begin{cases} 1 & \text{if } MSE \leq \varepsilon + e' \\ 0 & \text{otherwise} \end{cases}$$

where MSE is the mean absolute error made by an FOT model as defined before, ε is again a user-defined threshold (default $\varepsilon = 0.01$), e' is the mean of squared noise added into training data, i.e. the error of an FOT with optimal parameter specifications.

The experimental results are shown in Figure 3.24 according to different ε values. The results are very similar to the previous noise-free scenario, but as the complexity grows, the noise has a negative influence on the success rate, that is, the success rate starts to deteriorate slightly. Notice that after adding noise in the training data, the calibration process needs relatively more training time comparing of synthetic, noise-free training data.

3.4.4 Discrete Data

In this experiment, categorical data are considered for the calibration of FOTs. Originally an FOT can only give numeric outputs, in order to transform the numeric output of FOTs into discrete one, several "cut points" are used here to divide the unit interval into several regions, each of them corresponds a discrete output. In this experiment those cut points are randomly generated under the uniform distribution in the unit interval, apparently it needs $k-1$ cut points to generate k discrete outputs. In the calibration phase, the ES are set to tune these cut points just considering them as constraint parameters, this can be easily done in ES.

In contrast to the case of numeric outputs, the loss function MSE is newly defined here for FOTs with categorical outputs:

- **Mean squared error** : Given an FOT $F(x,\hat{\theta})$ with categorical output, the mean squared error of $F(x,\hat{\theta})$ is defined as:

$$MSE := \frac{1}{N} \sum_{(x_i,y_i)\in D} \gamma(\hat{y}_i, y_i)^2$$

where N is the size of training data D, $\hat{\theta}$ is the parameter specification of F,

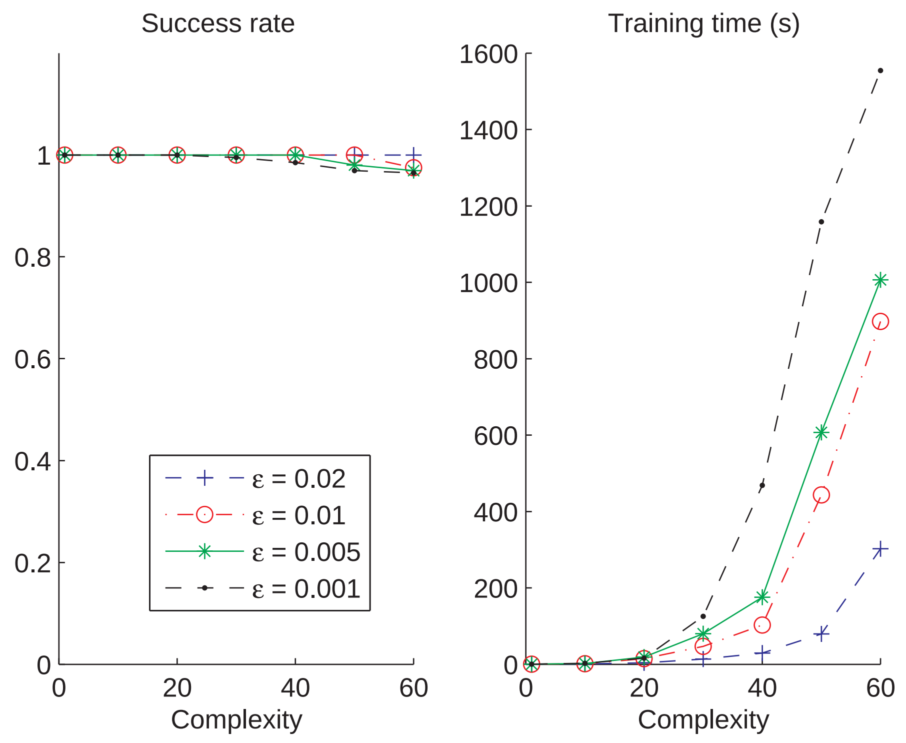


Figure 3.24: Experimental results on noisy data

$\hat{y}_i = F(x_i, \hat{\theta})$ and γ is the all-threshold loss defined before (see Equation 3.1.3).

Correspondingly, the *Suc* in this case is updated with newly defined *MSE*. The experiments with different number of outputs ($k = 2, k = 3$ and $k = 4$) have been carried out and shown in Figure 3.25.

Again the success rate on different ks is very high and only starts to deteriorate slightly as complexity increases. The calibration problem seems to become harder for a larger k, namely a larger number of categories, as one can see the increased runtime for $k = 4$.

3.4.5 Real Data

The goal of the study on real data is to compare FOTs with other model classes in a learning (classification) context, in order to show that it can be usefully applied in those cases, where expert knowledge about the qualitative structure of an evaluation scheme is available. In this section, let us consider an interesting type of rating problem, namely to evaluate strategic situations in the game of poker[2]. More specifically, we considered the pre-flop phase (two cards at first hand) in the Texas Hold'em variant (see [Skl03] for general introduction), where a player can take one of two possible actions: *Raise* and *call* [3]. As a player will normally tend to raise in situations that appear to be favorable and call in less favorable ones, these two actions can be considered as ordinal ratings, with *call*$\prec$*raise*, i.e. assuming that a player can evaluate the first hand by assign a numeric rating, then one will take *raise* in case of a high rating, and take *call* otherwise. From a machine learning point of view, the problem can obviously be seen as a binary classification task: Predict whether, in a given situation, a poker player will raise or call.

[2] Poker currently enjoys a great popularity in artificial intelligence research, especially in machine learning [FK01].

[3] Actually one can also use "fold" to quit the game, however in this case the card information is not recorded, thus not interested here.

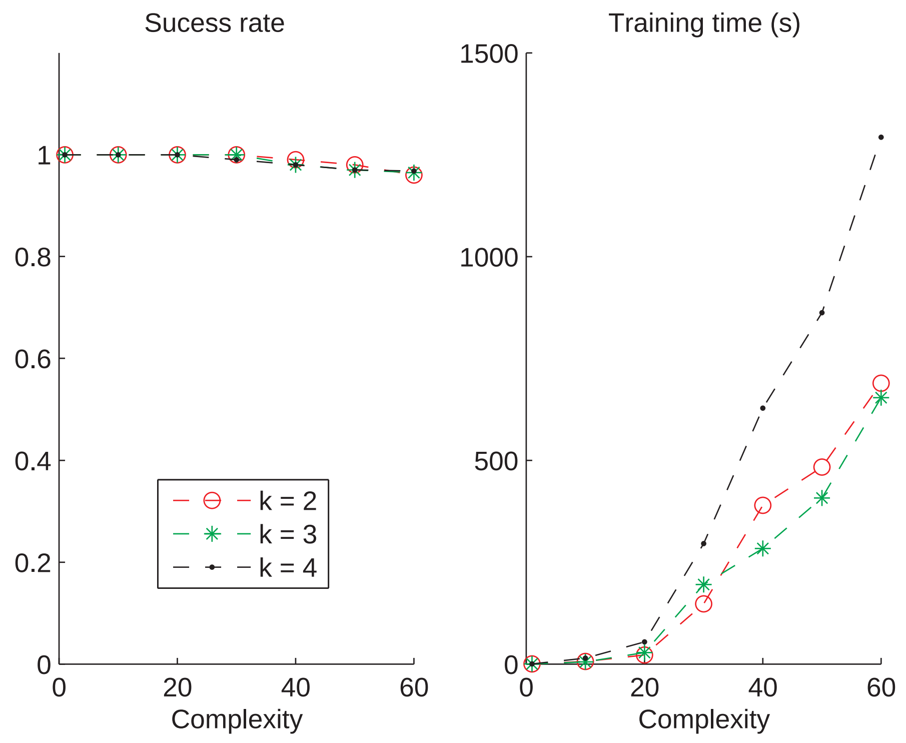


Figure 3.25: Experimental results on discrete data

The poker data is collected by the university of alberta computer poker research group[4], which contains currently over ten million records of information about poker plays, such as number of players, cards on board, and so on. According to our poker expertise, a relative small dataset is extracted from the original dataset, which contains 71732 instances related to the our task in this thesis. Furthermore, six basic evaluations are built with the help of our poker expertise, they are:

1. **cardvalue** : The quality of the first hand (with two cards) hold by a player. According to the Sklansky table[5], all possible first hands are categorized into nine levels, a smaller level corresponds a higher quality of the first hand. To modulate this input value, we define the fuzzy set of "strong hands" with a membership function:

$$(c_1, c_2) \to r + \frac{(1-r)S(c_1, c_2)}{8}$$

   where c_1 and c_2 express the first two cards, $S(c_1, c_2)$ is the Sklansky level, and $r \in [0,1)$ is a parameter to be learned.

2. **street** : The probability to extend the current hand to a street, using the (yet unknown) three cards that follow. The fuzzy set modulating this input feature is defined by:

$$(c_1, c_2) \to \frac{p(c_1, c_2)}{max_{c,c'} p(c, c')}$$

   where $p(c_1, c_2)$ is the probability to extend (c_1, c_2) to a street and defined by:

$$p(c_1, c_2) := \frac{|\{(x,y,z)|(x,y,z) \in \mathscr{C}, street(x,y,z,c_1,c_2)\}|}{|\mathscr{C}|}$$

   where $\mathscr{C}$ is the set of potential remaining three cards, which can build a *hand*

[4] http://games.cs.ualberta.ca/poker/IRC/
[5] http://www.onlinepokercenter.com/articles/poker_strategy/sklanskys_hand_rankings.php

together with the two cards at first hand, $street(x_1,\ldots,x_5)$ indicates whether $x_1,\ldots,x_5$ can build a street or not.

3. **flush** : The probability to build a flush based on current hand. Since a flush depends only on the color of cards, a fuzzy set modulating this input feature is defined as:
$$(c_1,c_2) \rightarrow \begin{cases} 1 & \text{if } c_1 \text{ and } c_2 \text{ have same color} \\ r & \text{otherwise} \end{cases}$$
where $r \in [0,1)$ is again a parameter to be learned.

4. **pair** : The probability to build a pair on current hand, which is simply defined as:
$$(c_1,c_2) \rightarrow \begin{cases} 1 & \text{if } c_1 \text{ and } c_2 \text{ only differ in color} \\ r & \text{otherwise} \end{cases}$$
with the parameter $r \in [0,1)$ to be learned.

5. **#player** : Since the best action also depends on the number of players involved in the game, the larger the number of players, the smaller the chance to win the game, thus the more probability to “call". Assuming that there are maximal 12 players, we define the modulating fuzzy set by:
$$n \rightarrow 1 - \frac{r(n-2)}{10}$$
with a parameter $r \in [0,1)$ to be learned and n the actual number of players.

6. **position** : From the strategic point of view, sitting at the end of the table is better than sitting at the beginning, namely the later to play, the better a position is. A fuzzy set of “good positions” is therefore defined by:
$$p \rightarrow r + \frac{p(1-r)}{n}$$
with the parameter $r \in [0,1)$ to be learned and p the actual position of player, n the actual number of players.

According our modest poker expertise, an FOT is designed based on previous attributes to model a utility function on the action attribute, as Figure 3.26 shows. Needless to say, the features that we used will not completely determine the action of a poker player, the real poker game is more complicated than in this experiment here. Instead, a player may consider further aspects, different players have different strategies (e.g. bluffing plays an important role in poker), eventually some randomness will also remain. As a result, the poker data can be considered as extremely noisy, which makes the prediction on poker data very difficult from a learning point of view.

To make a fair comparison with state-of-the-art algorithms, two well-known algorithms are taken from tree-based and rules-based classifier packages of the Weka package [FHT97] (*C*4.5 and *Ripper*). Table 3.5 lists these two algorithms and achieved accuracy [6] and complexity[7] on the poker data, with respective standard deviations. As one can see, the poker data is indeed very hard to be learnt, two algorithms achieve

[6] The accuracy is achieved by 10-fold cross-validation over all training data at an average.

[7] For rule-based algorithm, the complexity is the number of rules used in the model; for tree-based algorithm, the complexity is the number of nodes involved in the tree-like model.

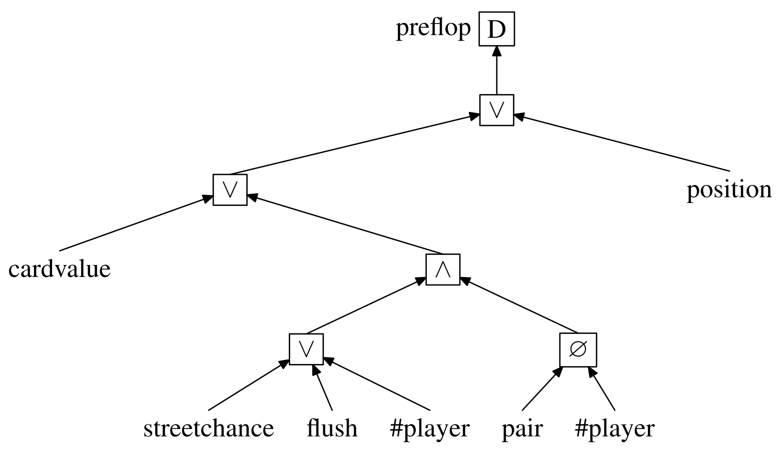


Figure 3.26: An FOT for rating actions on first hand in the poker game

until about 68.8% accuracy, in which the C4.5[8] usually returns the better accuracy than JRip [9]. On the other hand, the models on these algorithms are quite complex and difficult to be understood, even the simplest model learnt by JRip contains already about sixteen rules, although the length of single rule is still not considered. For a poker player, this kind of model is of course not easy to be digested. The FOT model on the poker data, in contrast, is very straightforward and intuitive, alone the interpretability of FOT has shown its advantage.

Table 3.5: Reference algorithms on the poker data

Name	Category	Accuracy(%)	Complexity
JRip	Rule-based	68.60 ± 0.45	15.01 ± 3.07
C4.5	Tree-based	68.84 ± 0.48	168.06 ± 17.60

Following, we investigate the accuracy of our FOT model on poker data. To do this, we randomly extract a finite number of training data, say T, and use the rest data for purpose of testing. For each T, the reference algorithms and our FOT model are provided a training data with exact T examples, and the accuracy is measured by applying models on remaining testing data (with $71732 - T$ instances). In this way, the experiment would be repeated ten times, the results in terms of averaged accuracies as a function of the number of training data T are shown in the upper half of Figure 3.27, as well as averaged accuracies with standard deviations in the lower half of Figure 3.27 (JRip is omitted for the sake to clearness).

One of the most interesting aspects of the results concerns the fact that FOT significantly outperforms the other methods, if relatively few training data is available. As the provided training data increases, this superiority decreases and finally disappears. In fact, the improvement of FOT is drastic for the first few training examples, whereas the rest of the examples does not contribute very much. Besides, the standard deviations of the accuracy is significantly smaller for FOT than for the other models. To explain this finding, note that our FOT has a very strong inductive bias, due to the fact that it has a fixed, predetermined structure. Compared to this, the other models are much more flexible, so that they can adjust its structure according the mount of available training data, which reflects in the increased complexity of the learnt models (see Figure 3.29). It is well-known that, when learning models from data, a high flexibility of a model comes along with a risk of *over-fitting*, which is especially high for small data sets and becomes less severe for larger ones. Therefore, a strong inductive bias is especially useful if data is not abundant. However, at least if the bias is not perfect, if may also become hindering, namely to extract an optimal hypothesis even from a very rich model class, when enough data is available; in this case, the bias may

[8] In Weka it corresponds the J48 class.
[9] An implementation in Weka for Ripper.

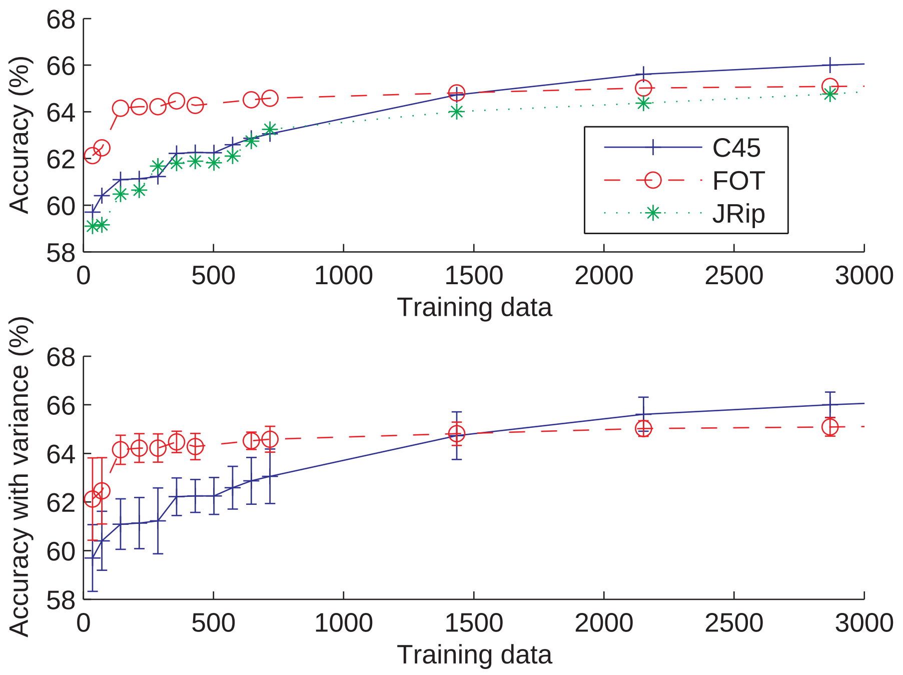


Figure 3.27: Experimental results on poker data (accuracy and variance)

produce a problem of *under-fitting*. Exactly this phenomenon seems to be responsible for the results that we obtained. In order to illustrate this effect, the accuracies on training and test data for these approaches are plotted in Figure 3.28, in which the learnt models are applied to classify the training and test data respectively. These results again confirm that our approach is especially useful if reliable background knowledge is available, as it allows one to formalize and exploit such knowledge in a convenient way, while alternative approaches may otherwise be preferable.

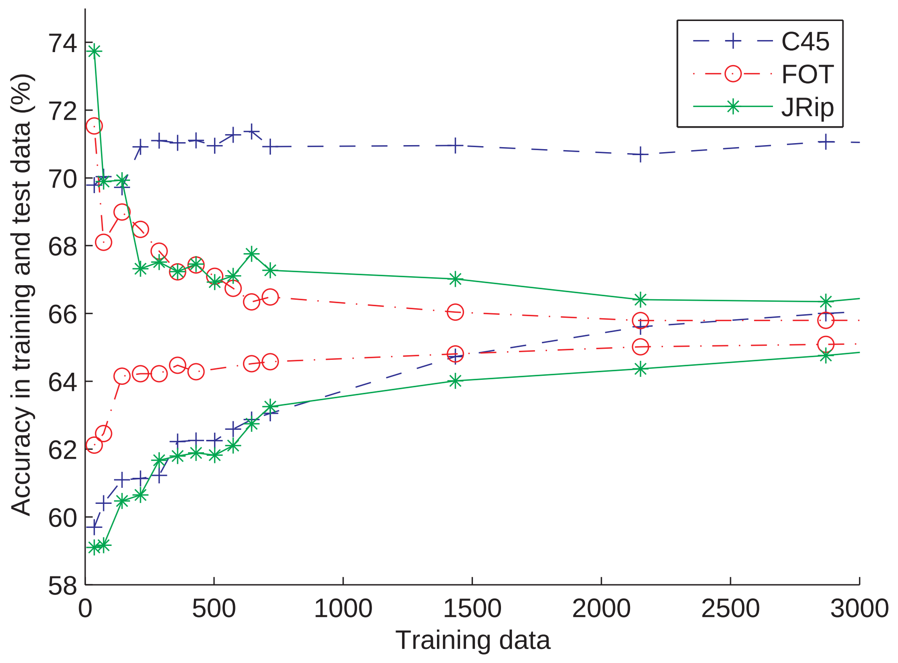
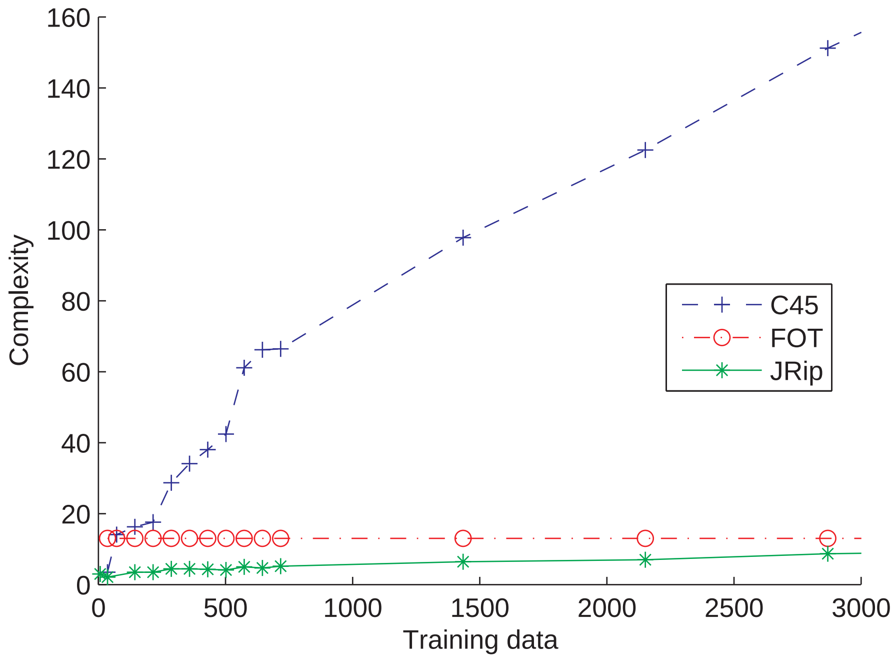


Figure 3.28: Experimental results on poker data (accuracy in training and test data)

To measure the interpretability, the complexity of models used in C4.5 and our FOT is taken into account in this case, since both have a tree-like structure, the complexity of models is defined as the number of nodes in the tree structure. Though models in JRip have different structure as our FOT, we consider the complexity of JRip as the number of rules involved in models, but we should keep in mind that the rules in

JRip may have still various lengths. The complexities for two reference algorithms and our FOT as a function of the number of training data T are shown in Figure 3.29. Note that the structure of FOT is fixed in this case, so that its complexity is a constant line here (complexity = 13).

Figure 3.29: Experimental results on poker data (complexity of models)



Let us again bring up the aspect of interpretability, which we consider as another strong advantage of our approach. In fact, even though the C4.5 models (as well as JRip classifiers) achieve a slightly higher classification rate if enough training data is available, these models are quite complex and difficult to understand. The size of the decision trees tends to increase with the size of the training data and can easily contain as many as 60 interior nodes; besides the splitting conditions are often non-intuitive. While such models might still be acceptable to predict actions in Poker, interpretability becomes much more an issue in critical applications, such as quality control or classification in medicine. In these fields, a human expert may not be willing to trust in a decision tree model with 60 interior nodes.

3.5 Conclusions

In this chapter we address the optimization problem of specifying the parameters involved in an FOT and several considered calibration techniques respectively. The empirical evaluation with these calibration techniques has been investigated in this chapter, which shows that the evolution strategies delivers a satisfied optimization performance in terms of accuracy and running time, even on noisy and discrete data. As applied to a real problem, the FOT model shows its superiority in special situation against several well-known techniques, where an FOT model has strong inductive bias, which becomes especially useful in the case that provided data is not abundant. On the other hand, maybe more important argument to use FOTs, an FOT model has the excellent interpretability, whereas many purely data-driven techniques are usually too complicated to be understood by human being.

4 Minimizing Evaluation Costs for Fuzzy Operator Trees

In this chapter we address a cost issue when applying FOTs, namely how to minimize evaluation cost on FOT, if the basic evaluations are associated with finite costs. The motivation and related definitions for this problem are introduced in Section 4.1, while Section 4.2 is devoted to the related considerations and algorithms.

4.1 Problem Statement

Let us start with the motivation for evaluation cost minimization on FOTs in Subsection 4.1.1, then we give related definitions in Subsection 4.1.2. An illustrative example is shown in Subsection 4.1.3

4.1.1 Motivation

The problem of minimizing evaluation cost on FOT arises regarding following two observations:

1. How precise do we need an assessment? An FOT is supposed to assess an object by assigning a precise utility degree in the unit interval, which is special useful and necessary for several applications, such as object comparison, ordering, etc. But in other cases, user may not always need a precise assessment, but have different requirements on precision level. Figure 4.1 illustrates an example on product evaluation, where the precision level 1 can be used to detect "defective" products by setting a threshold on 0.9, so a product will be labeled as *NIO*, if the assessment reported by FOT lies in the interval $[0, 0.9)$, otherwise it is labeled as *IO*. In this way, one can distinguish the quality of product roughly and classify the products into two classes. In industrial area, this might be very useful for early detection of serious errors in the production process. On the other hand, a more precise level 2 with another four thresholds is to determine "classes" of approved products, just like in the lower half of Figure 4.1 shows. A more detailed classification on "approved" products might be very useful for several tasks, for example, different prices can be set down based on this classification, or assembling of components from different "classes" can be arranged, etc. In a word, we do not always need precise assessments, a rough classification is enough in many cases.

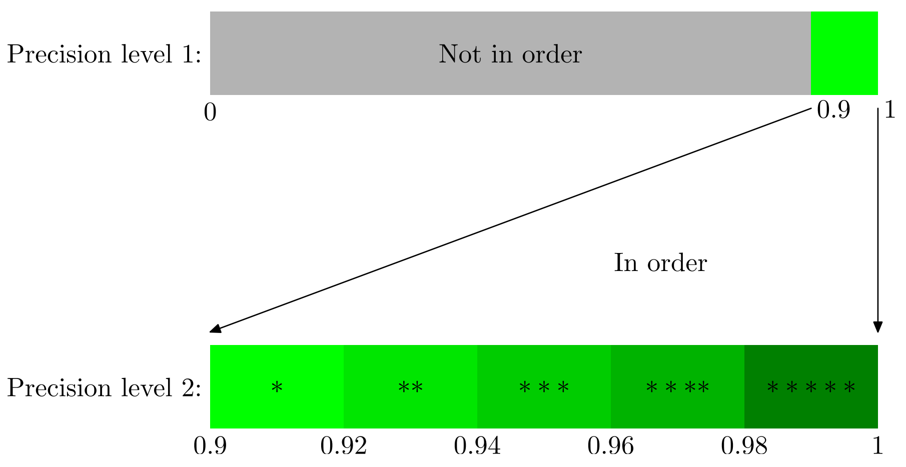


Figure 4.1: Precision levels of assessment on FOT

Classification instead of precise assessment is the basic motivation for the works in this chapter. Previously, in order to get a precise assessment of object based on an FOT, measurements for all basic evaluations in the leaf nodes must be carried out generally. However if the goal of quality assessment turns to classify an object into classes regarding to quality threshold(s), not all measurements have to be carried out, since the classification of object might be determined based on just part of measurements, in this case the remaining measurements are superfluous in the sense that they can not change the final classification of object, but only refine the assessment made by already evaluated measurements. This motivates us to analyse the evaluation order according to given precision levels.

2. In practice, the measurement of basic evaluations on an FOT is always associated with an evaluation cost, for example considering the car selection problem introduced in Section 2.4.2.3, the basic evaluation (number of) "Doors" is trivial, while for another basic evaluation "Safety", it might be necessary to arrange several crash experiments to determine the safety factor. Of course from a producer's point of view, a minimization of overall evaluation cost is desired.

The goal of minimizing evaluation cost on FOTs is how to arrange the measurements (since the basic evaluations of an FOT are built based on measurements though associated fuzzy sets, in the following text, we simply use the term *basic evaluation* to refer a measurement and followed basic evaluation through associated fuzzy set instead distinguishing both terms strictly), in order to determine the class of object with given precision level, the objective criterion for such arrangement is the evaluation cost on carried basic evaluations until the classification of object is determined, generally the less evaluation cost needed, the better is an arrangement.

4.1.2 Formal Definition

For further discussion, we give related definitions and terminologies in this section. Given an FOT $F(x_i, x_2, \dots, x_n)$ with n basic evaluations, we assume that each basic evaluation (in the unit interval $[0,1]$) follows a finite distribution (denoted as $D(x_i)$), which is represented in the form of a *probability density function*, so that each basic evaluation can be seen as a random number under a fixed distribution. Furthermore, the evaluation cost of the i-th basic evaluation x_i is denoted as $c(x_i)$, which is assumed to be normalized into the unit interval $[0,1]$. For the sake of clearness, we use the term *cost* for the evaluation cost of basic evaluation and leave the term *evaluation cost* for further purpose, as introduced later.

Definition 4.1 (Evaluation Plan)**.** An *evaluation plan* P (*plan* for short) on an FOT $F(x_1,x_2,\ldots,x_n)$ is defined as a permutation of indices of x_i from $\{1,\cdots,n\}$, formally, P is a bijective mapping:

$$P:\{1,2,\ldots,n\}\rightarrow\{1,2,\ldots,n\}$$

An evaluation plan P gives the sequence, in which the basic evaluations should be carried out, $P(i)$ denotes the index of basic evaluation carried in the i-th step according to an evaluation plan P. According to an evaluation plan, the basic evaluations in an FOT would be carried out sequentially. In following text, we shall use $F^{(i)}$ to denote an FOT F after the i-th evaluation steps, namely the first i basic evaluations indicated by P have been carried out, and their outputs are known, whereas the remaining basic evaluations are still unknown. In particular, $F^{(0)}$ denotes an FOT F, in which none of its evaluations is carried out. Furthermore, we use $F^{(i)}(x)$ to denote the output of $F^{(i)}$, if the remaining basic evaluations are assigned with x.

Definition 4.2 (Quality Interval)**.** Given an FOT $F(x_1,x_2,\ldots,x_n)$ and a plan P, an *quality interval* (*interval* for short, denoted as $[F^{(i)}]$) after the i-th evaluation step is defined by:

$$[F^{(i)}] := [\underline{F^{(i)}},\overline{F^{(i)}}]$$

where $i\in\{0,1,\ldots,n\}$ and

$$\underline{F^{(i)}} := \min_{x\in\mathbb{X}^{n-i}}(F^{(i)}(x))$$

$$\overline{F^{(i)}} := \max_{x\in\mathbb{X}^{n-i}}(F^{(i)}(x))$$

in which $\mathbb{X}^{n-i}=[0,1]^{n-i}$ is the domain of the remaining $n-i$ basic evaluations to be carried out.

Note that an interval indicates the range of possible output of an FOT after some basic evaluations have been carried out. Apparently, as more basic evaluations are carried out, the certainty of final output of an FOT increases, which corresponds a smaller interval.

Proposition 4.3. *Given an FOT $F(x_1,x_2,\ldots,x_n)$ and a plan P, it holds:*

- $[F^{(0)}]=[0,1]$
- $[F^{(i)}]\subseteq[0,1]$, $\forall i\in\{0,\ldots,n\}$
- $[F^{(i)}]\subseteq[F^{(i-1)}]$, $\forall i\in\{1,\ldots,n\}$
- $\forall x\in\mathbb{X}^{n-i}: F^{(i)}(x)\in[F^{(i)}]$, $\forall i\in\{0,\ldots,n\}$
- $\forall y\in[F^{(i)}]\ \exists x\in\mathbb{X}^{n-i}: F^{(i)}(x)=y$, $\forall i\in\{0,\ldots,n\}$
- $\underline{F^{(i)}}=F^{(i)}(0^{n-i})$ *and* $\overline{F^{(i)}}=F^{(i)}(1^{n-i})$, $\forall i\in\{0,\ldots,n\}$
  where 0^{n-i} *(*1^{n-i}*) indicates the case that all remaining* $n-i$ *basic evaluations have output of* 0 *(*1*), namely:*

$$x_{P(j)}=0,\ \forall j\in\{i+1,\ldots,n\}$$

or

$$x_{P(j)}=1,\ \forall j\in\{i+1,\ldots,n\}$$

The proofs of Proposition 4.3 are trivial and omitted here.

Definition 4.4 (Quality Threshold)**.** To assess an object o on an FOT F, a *quality threshold* (*threshold* for short) T is a real number in $[0,1]$, which expresses the users interest whether the assessment of o lies over or under T.

The term threshold is defined here to express a precision level, for example, the precision level 1 in Figure 4.1 can be built with a threshold $T = 0.9$. Notice that several thresholds build a multi-classes evaluation, namely given a set of thresholds $T = \{T_1, T_2, \dots, T_n\}$ $(0 \leq T_1 < T_2 < \dots < T_n \leq 1)$ and an object o, following classes can be built:

$$\begin{cases} \text{class } 0 & \text{if } 0 \leq F(o) < T_1; \\ \text{class } 1 & \text{if } T_1 \leq F(o) < T_2; \\ \dots & \\ \text{class } n-1 & \text{if } T_{n-1} \leq F(o) < T_n; \\ \text{class } n & \text{if } t_n \leq F(o) \leq 1; \end{cases}$$

Definition 4.5 (Quality Approved)**.** Given an FOT $F(x_1, x_2, \dots, x_n)$, an integer $i \in \{1, \dots, n\}$, a plan P and a threshold T, the quality of an object o is called *approved* at the i-th step, if after the i-th basic evaluation being carried out, the quality of o is ensured to be at least T, called *positively approved* (denoted as $Q_{\geq}(i,F,P,T) = 1$), or under T, called *negatively approved* ($Q_{<}(i,F,P,T) = 1$), otherwise the quality of an object o is called *not approved* at the i-th step ($Q_{\geq}(i,F,P,T) = Q_{<}(i,F,P,T) = 0$). Formally, we have:

$$Q_{\geq}(i,F,P,T) = \begin{cases} 1 & \text{if } [F^{(i)}] \subseteq [T,1] \\ 0 & \text{otherwise} \end{cases} \tag{4.1.1}$$

$$Q_{<}(i,F,P,T) = \begin{cases} 1 & \text{if } [F^{(i)}] \subseteq [0,T) \\ 0 & \text{otherwise} \end{cases} \tag{4.1.2}$$

where $i \in \{1, \dots, n\}$. Since $[F^{(i)}]$ is a continuous interval and bounded by $[0,1]$, Equation 4.1.1 and 4.1.2 can also be rewritten as:

$$Q_{\geq}(i,F,P,T) = \begin{cases} 1 & \text{if } \underline{F^{(i)}} \geq T \\ 0 & \text{otherwise} \end{cases}$$

$$Q_{<}(i,F,P,T) = \begin{cases} 1 & \text{if } \overline{F^{(i)}} < T \\ 0 & \text{otherwise} \end{cases}$$

Together the quality of an object o is called *approved* at the i-th step, denoted as $Q(i,F,P,T) = 1$, if it can be either positively or negatively approved, otherwise denoted as $Q(i,F,P,T) = 0$, that is:

$$Q(i,F,P,T) = sgn(Q_{\geq}(i,F,P,T) + Q_{<}(i,F,P,T))$$

with *sgn* is the signum function.

Definition 4.6 (Evaluation Cost)**.** Given an FOT F, a plan P and a threshold T, the *evaluation cost*, written as $E(F,P,T)$, is the expected sum of costs of basic evaluations, which contribute to quality approving. Note that $Q(i,F,P,T)$ is a random variable based on the first i basic evaluations according to P. Formally, the evaluation cost is defined:

$$\begin{aligned} E(F,P,T) &:= \sum_{i=1}^{n} c(x_{P(i)})(1-p(Q(i-1,F,P,T)=1)) \\ &= \min_{*\in\{\geq,<\}} \left(\sum_{i=1}^{n} c(x_{P(i)})(1-p(Q_*(i-1,F,P,T)=1)) \right) \end{aligned} \tag{4.1.3}$$

where $p(\cdot)$ denotes the probability function.

As we use $Q(i,F,P,T)$ to mark whether the quality of to be evaluated object is approved or not after the i-th evaluation step, the evaluation cost is an accumulative cost of those basic evaluations, which are responsible for quality approving. Actually, the cost of i-th basic evaluation would be counted, only if the quality is still not approved before carrying the i-th basic evaluation, namely if $Q(i-1,F,P,T)=0$. Otherwise, in the case that the quality is already approved before the i-th evaluation step, the basic evaluations in the i-th step and after that are treated as superfluous, and not counted into evaluation cost.

Notice that in the worst case, the evaluation cost could be maximal, namely the sum of costs of all basic evaluations, since in that case, the quality can only be approved, after all basic evaluation are executed. Otherwise, if we can find an evaluation plan, in which the quality can be approved as "soon" as possible, then the total evaluation cost can be decreased. Therefore, the challenge of minimizing evaluation cost in this chapter is to find an optimal evaluation plan, which can minimize the evaluation cost, under the condition that all involved basic evaluations are independent and follow finite distributions.

Definition 4.7 (Optimal Evaluation Plan)**.** Given an FOT F and a threshold T, let us assume that $\mathbb{P}$ is the set of possible evaluation plans on F, then an *optimal evaluation plan P^** is defined as:

$$\begin{aligned} P^* &:= \arg\min_{P\in\mathbb{P}} E(F,P,T) \\ &= \arg \min_{p\in\mathbb{P},*\in\{\geq,<\}} \left(\sum_{i=1}^{n} c(x_{P(i)})(1-p(Q_*(i-1,F,P,T)=1)) \right) \end{aligned} \tag{4.1.4}$$

To this end, the goal of minimizing evaluation cost on FOTs is to find an optimal evaluation plan P^*, which is expected with minimal evaluation cost.

Proposition 4.8. *The problem of minimizing evaluation cost on FOTs is NP-hard.*

Proof. As usual, in order to establish *NP*-hardness of the problem of minimizing evaluation cost, say $\mathscr{P}$, it suffices to demonstrate that a certain problem, which is known to be *NP*-hard, can be polynomially reduced to $\mathscr{P}$.

On the one hand, let us start from a well-known *NP*-hard problem: The minimization knapsack problem [KPP04, MT90], which is a minimization variant of the well-known $0-1$ knapsack problem [GJ79, GL79], let us refer the minimization knapsack problem as $\mathscr{P}^+$ later. In following, there are n kinds of projects $(I_1,I_2,\ldots,I_n)$, associated with each project I_i are two positive number $p(I_i)$ and $c(I_i)$. If $S\subseteq\{1,2,\ldots,n\}$ is a subset of indices of projects, then the problem of $\mathscr{P}^+$ is to find an optimal subset of $\{1,2,\ldots,n\}$, say S^*, which is defined by:

$$S^* := \arg \min_{S\subseteq\{1,\ldots,n\}} \left(\sum_{j\in S} c(I_j) | \sum_{j\in S} p(I_j) \geq d \right) \tag{4.1.5}$$

with a positive number d. According to [DK08], we can interpret $c(I_j)$ as the costs for realization of the project I_j and $p(I_j)$ as the money effect from its realization. This problem is to find a set of projects the joint realization of those would provide the budget revenue not less than d under minimal total costs for realization of this set.

On the other hand, we try to give an reformulation from $\mathscr{P}^+$ to the problem of minimizing evaluation cost, referred $\mathscr{P}$ as later. Let us consider a special case of $\mathscr{P}$ firstly, in which an FOT F has n basic evaluations, any basic evaluation, say at position i, is associated with a fixed number x_i ($x_i \in [0,1]$) as output and cost $c(x_i)$, that is, all basic evaluations are not considered as random variables, but have fixed outputs. Beside this, there is only one aggregation operator $M_{OWA}(\frac{1}{n},\dots,\frac{1}{n})$ in F, as Figure 4.2 shows. Moreover, considering the parameter specification in M_{OWA}, one can follow that:

$$\underline{F}^{(i)} = \frac{1}{n}\sum_{j=1}^{i} x_{P(j)}$$

According to Definition 4.1.4, it is enough to show the *NP*-hardness of a sub-problem of $\mathscr{P}$, namely to find an optimal evaluation plan for positive approving, denoted as $\mathscr{P}'$. The problem of $\mathscr{P}'$ is to find a plan P^*, which is defined by:

$$\begin{aligned} P^* &:= \arg\min_{p\in\mathbb{P}} \left(\sum_{i=1}^{n} c(x_{P(i)})(1 - p(Q_{\geq}(i-1,F,P,T) = 1)) \right) \\ &= \arg\min_{P\in\mathbb{P}} \left(\sum_{i=1}^{k} c(x_{P(i)}) | k \in \{1,\dots,n\}, \sum_{i=1}^{k} x_{P(i)} \geq n \cdot T \right) \end{aligned} \tag{4.1.6}$$

with a threshold T, k is the number of evaluation steps, so that after the k-th evaluation step, the quality can be approved positively. Notice that p disappears at the end of Definition 4.1.6, because in this special case, all basic evaluations have fixed outputs. As the outputs of basic evaluations are fixed in this special case, the problem of $\mathscr{P}$ turns to find a subset of basic evaluations with minimal evaluation costs, so that the quality can be positively approved.

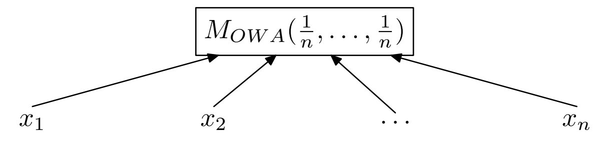


Figure 4.2: A special case for minimizing evaluation cost problem

Comparing Definition 4.1.5 and 4.1.6, it is not difficult to figure out the similarity of the problem $\mathscr{P}^+$ and $\mathscr{P}'$. As a result, $\mathscr{P}^+$ can be reduced to $\mathscr{P}'$ through several reformulations, which are listed in Table 4.1. For any valid solution for $\mathscr{P}^+$, say S, we can simply construct a valid solution (denoted as P) for $\mathscr{P}'$ by, for instance, concatenating the elements in S and $\{1,\dots,n\} \setminus S$.

Reversely, given a valid solution for $\mathscr{P}'$, say P, let us locate the number k in Definition 4.1.6 at first, by using:

$$k = \min\{j | 1 \leq j \leq n, \sum_{i=1}^{j} x_{P(i)} \geq n \cdot T\}$$

Then a valid solution for $\mathscr{P}^+$, say S, can be constructed as:

$$S := \{P(i) | 1 \leq i \leq k\}$$

Apparently, the reformulation from $\mathscr{P}^+$ to $\mathscr{P}'$ can be achieved in polynomial time.

Finally, since $\mathscr{P}^+$ is *NP*-hard, so it follows that $\mathscr{P}'$ and $\mathscr{P}$ is also *NP*-hard, the more general problem of minimizing evaluation cost is of course *NP*-hard.

Table 4.1: Reformulation from the minimization knapsack problem to the problem of minimizing evaluation cost.

Nr.	In $\mathscr{P}^+$	In $\mathscr{P}'$	Remark
1	I_i	i-th basic evaluation	Project $\Rightarrow$ Basic evaluation
2	$p(I_i)$	x_i	Money effect $\Rightarrow$ Output of basic evaluation
3	$c(I_i)$	$c(x_i)$	Cost of realization $\Rightarrow$ Evaluation cost
4	d	$n \cdot T$	Budget revenue $\Rightarrow$ Threshold
5	S	P	Set of projects $\Rightarrow$ Permutation of basic evaluations

$\square$

4.1.3 An Illustrative Example

An illustration example based on simplified car selection problem from Section 2.4.2.3 is shown in Figure 4.3, where every basic evaluation (at leaf node) has a fixed cost denoted with c and for an exemplary object o, its basic evaluations are expressed in an input vector X. For the sake of clearness, we assume that the norms involved in this example are the minimum t-norm, the maximum t-conorm and the arithmetic mean respectively. To get the precise utility of o, all basic evaluations in X are required usually, then the utility of o could be read at the root node (marked with red in the parenthesis), as well as all the intermediate results.

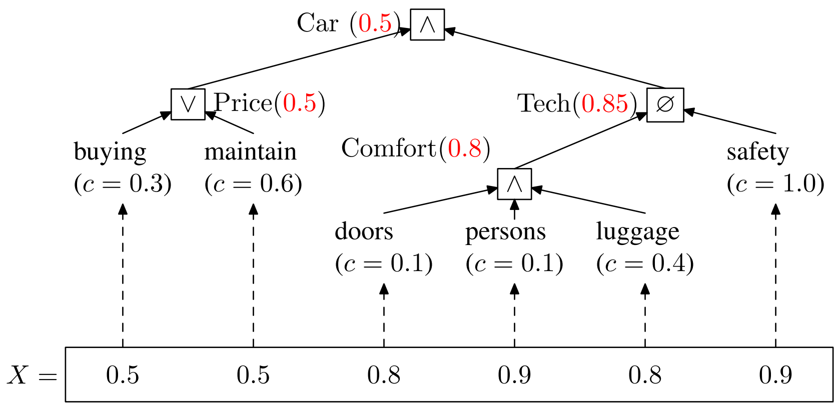


Figure 4.3: Evaluation cost for car selection example

But if one interests in whether o can be positively approved regarding a given T, not all basic evaluations in X are required to determine the quality of o. Let us assume a threshold $T = 0.9$ is given, which can be read as: All objects are classified into two classes according to their quality degrees, the "positively approved" class (quality degree over or equal 0.9, indifferent among $0.91, 0.95$ and 1.0) and the "negatively approved" class (quality degree under 0.9). Now we are considering two simple evaluation plans:

$$P_1 := [1,2,3,4,5,6]$$
$$P_2 := [6,5,4,3,2,1]$$

namely the six basic evaluations involved in this example would be measured one by one beginning from "Buying" rightwards under P_1, and in reverse direction under P_2. As the evaluation process under both plans are plotted in Figure 4.4 to 4.5, it is not difficult to see that the evaluation cost denotes 0.9 $(= 0.3 + 0.6)$ under P_1, namely the sum of costs of "Buying" and "Maintain", since the "Price" factor based on them causes the overall quality of car to be limited in an interval $[0, 0.5]$ due to the boundary rule of t-norm, which means o is approved negatively in any way, no matter the remaining basic evaluations (denotes using "?"). On the other side, the evaluation cost under P_2 turns to be $1.6 (= 0.1 + 0.1 + 0.4 + 1.0)$, since the quality can only be approved until the first four basic evaluations are carried out ("Doors","Persons","Luggage","Safety"). Obviously for this object o, the evaluation cost under P_1 is less than that under P_2, actually P_1 is one of optimal evaluation plans on o in this case.

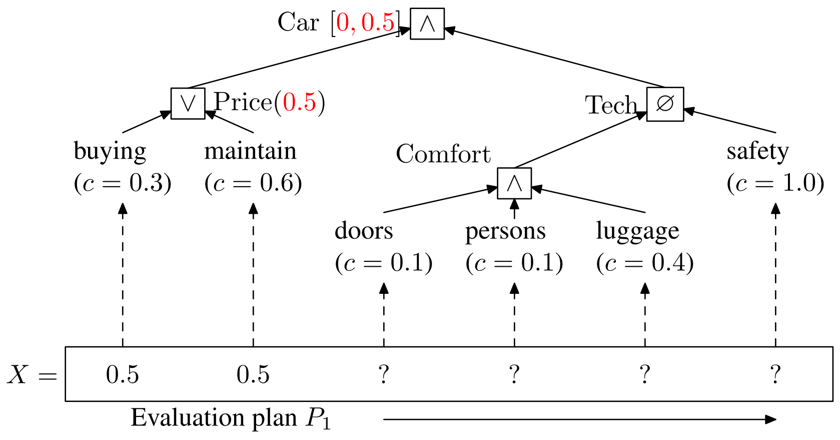


Figure 4.4: Evaluation plan P_1 for car selection example

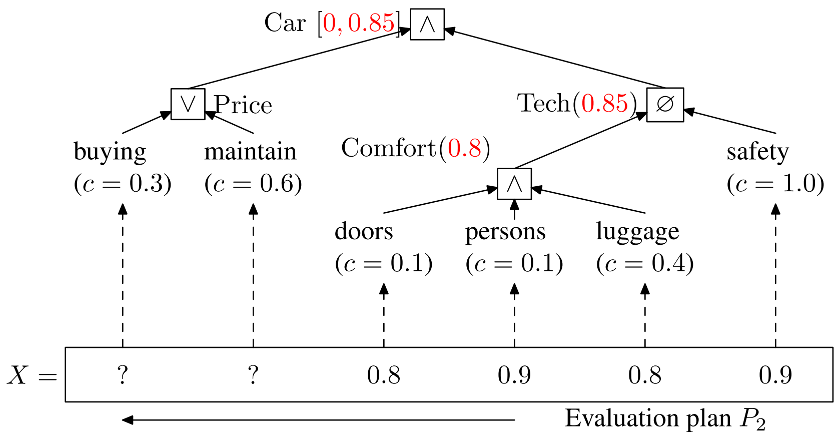


Figure 4.5: Evaluation plan P_2 for car selection example

4.1.4 Background and Concepts

In this thesis, we shall consider two different ways to determine an evaluation plan:

- **Static plan** : In this case, the whole evaluation plan would be determined in one step, i.e. before the real evaluation process begins, the whole evaluation plan is determined and keeps fixed in the real evaluation process, which is independent of the instances to be evaluated. Obviously, there are $n!$ candidate evaluation plans for an FOT with n basic evaluations, the task here is to determine the "optimal" one among those candidates, we shall discuss this case in Section 4.2.4;
- **Online plan** : Another way is that instead of deducing a complete evaluation plan at one time, one can select one basic evaluation each time, then the selected basic evaluation would be carried out, these two steps would be repeated until the quality can be approved. In this case, each instance has its own evaluation plan, so this kind of plans is not fixed, but dynamic, more details are given in Section 4.2.5

Obviously, the background of determining a static plan is *decision theory* [How68, KR76, Ana93, Cle95], as determining an optimal evaluation plan can be considered as a decision process generally. Decision theory provides a principled framework for decision making under uncertainty for a rational individual by using statistical tools, probability theory, etc. There are a lot of related works to the problem discussed in this chapter: First of all, the basic evaluation is associated with a cost function, the more basic evaluations are executed, the more certainty about the quality of object and the more has to be paid for that. This problem has been researched under the concept of *Value of information* (VoI), which has long tradition in decision analysis and gained great interests in different research areas [How66, DS05]. The general idea to solve this kind of problems is to maximize the expectation of profit (or minimize cost), for which the crucial point is to set up the expectation function for profit (or cost).

On the other hand, to determine an online plan is a *sequential decision making* process [BS95, BSW89], which is a fundamental task typically faced by intelligent agent in a dynamical environment. Generally, the goal of a sequential decision making problem is to decide the sequences of jobs, tasks or actions, so that some predefined objective function can be optimized [SW78, JK02, Won80].

4.2 Estimating Optimal Evaluation Plan

Formally, the problem of minimizing evaluation cost is follows:

Given – an FOT F with n basic evaluations $(x_1, x_2, \ldots, x_n)$;
– a quality threshold T;
– positive costs for basic evaluations $c(x_i)$, $i \in \{1, \cdots, n\}$;
– distributions of basic evaluations $D(x_i)$, $i \in \{1, \cdots, n\}$;

Find – an optimal evaluation plan P.

Note that here we take several assumption for convenience:

- Only one threshold is considered here, so our interest is whether the quality of objects can be approved positively or negatively;
- The basic evaluations are independent of each other;
- The costs of basic evaluations are normalized into the unit interval $[0,1]$, in which the cost of each basic evaluation is scaled by dividing the maximal cost among all basic evaluations, so that the nearer a cost to 1, the more cost is associated with a basic evaluation. Since we are interesting in the sequence of basic evaluations to be carried out, which needs the overall evaluation cost as less as possible, but not the evaluation cost itself, the normalization will not change the order of basic evaluations to be carried out;
- Any basic evaluation follows a statistical distribution, which is represented with continuous probabilistic density function on the unit interval $[0,1]$.

4.2.1 The Basic Idea

Let us denote the set of all possible plans for an FOT $F(x_1, x_2, \ldots, x_n)$ by $\mathbb{P}$, for identifying an optimal evaluation plan P^*, an exhaustive search needs to check $|\mathbb{P}|$ $(= n!)$ plans, which becomes extreme time-consuming as the number of basic evaluations

n increases. As Proposition 4.8 indicates, the identification of an optimal evaluation plan is considered to be NP-hard, therefore, we try to approximate the optimal evaluation plan by using heuristic approaches in this section.

An optimal evaluation is expected to approve the quality of an object with little costs as possible, which can be interpreted in two ways: Firstly, the quality of an object should be approved as "soon" as possible, which corresponds how to tighten the interval according to predefined threshold quickly; secondly, the cost of selected basic evaluation should be kept as small as possible. Both aspects are important to determine an optimal evaluation plan, because a "quick", but "expensive" quality approving is not wished, neither a "cheap", but "slow" one. The basic idea to find an optimal evaluation plan here is to consider both aspects together by introducing a new measurement, which combines the cost of basic evaluation and its possible contribution to tighten the interval.

Nevertheless, as the previous example shows, the properties of aggregation operators involved in FOTs can be used to detect an optimal evaluation plan quickly, for instance, the boundary property of t-norm is useful for negatively approving, etc. Considering the special hierarchical structure of FOTs, we can analyse these properties of aggregation operators in a recursive manner, so that this problem can be divided into several sub-problems following the hierarchical structure of FOTs.

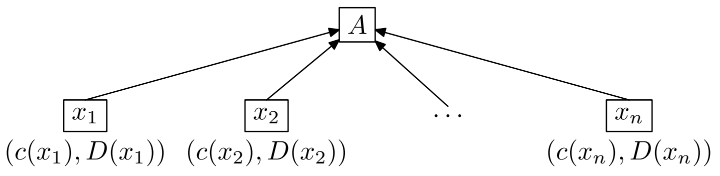
In the following discussion, we shall start with an analysis about the useful properties of aggregation operators in Subsection 4.2.2, then we discuss how to apply these properties upon the hierarchical structure of FOTs in Subsection 4.2.3. To this end, two algorithms are proposed to estimate an optimal evaluation plan in this section:

- a static algorithm decides a plan at the beginning of the evaluation process, and the resulted plan stays fixed afterward, we shall take a closer look in Subsection 4.2.4;
- an online algorithm tries to determine the evaluation plan during the evaluation process, so the resulted plan is different according to given object to be evaluated. This algorithm would be discussed in Subsection 4.2.5.

4.2.2 Role of Norms

Let us start with an FOT in a simple form, say F, which consists of a single aggregation operator (denoted as A) and n basic evaluations with the evaluation cost (c) and distribution (D) respectively, as Figure 4.6 illustrated.

Figure 4.6: An FOT in a simple form



To investigate the influence of different kind of aggregation operators, let us get more specification on an FOT in Figure 4.6, and assuming the aggregation operator A is a t-norm and there are only two basic evaluations, say x_1 and x_2. In this case there are only two possible evaluation plans, say $P_a = [1,2]$ and $P_b = [2,1]$, the question here is which plan can get the smaller expected evaluation cost? Due to the boundary property of t-norm, one can induce that:

$$x_1 < T \text{ or } x_2 < T \Rightarrow y < T$$

for any $T \in [0,1]$, independent of the parameter specification in the t-norm. If the first basic evaluation turns out lying under a given threshold T, the interval of FOT is tightened into $[0,T)$, since zero is the absorbing element for any t-norm. So, in this case, it is reasonable that we only try to approve the quality of FOTs negatively, whereas for positively approving, it is necessary to carry out all basic evaluations, which means the evaluation cost is in any case maximal. For negatively approving, we get the estimated evaluation cost under both plans (P_a and P_b) as follows:

$$E(F,P_a,T) = c(x_1) + c(x_2) * (1 - p(x_1 < T))$$

and

$$E(F,P_b,T) = c(x_2) + c(x_1) * (1 - p(x_2 < T))$$

for given threshold T, where $p(x_1 < T)$ denotes the probability that the basic evaluation on x_1 is lower than T. It is clear in this case that at least one basic evaluation has to be carried out in any case, but whether the other one is also necessary to be evaluated depends on the degree of basic evaluation carried before. As long as the first basic evaluation lies above the given threshold T, the other one needs to be evaluated in any way, which is independent of which t-norm is used here. To compare the evaluation costs under both plans, we give a new measurement in following text.

Definition 4.9 (Effective Cost)**.** Given a threshold T, the *effective cost* of a basic evaluation on x_i, written as $cp(x_i,T)$, is defined by:

$$cp(x_i,T) := \frac{c(x_i)}{p(x_i,T)}$$

where $c(x_i)$ is the cost of x_i, $p(x_i,T)$ is the probability of basic evaluation on x_i lies over or equal T in case of positive approving (called effective cost for positive approving), or under T in case of negative approving (called effective cost for negative approving).

The new measurement effective cost combines the two important aspects related to an optimal evaluation plan, namely the cost of basic evaluations and the probability of contribution for quality approving. An effective cost scales evaluation cost proportional to the probability, how the evaluation can distribute to quality approving, where the probability can be extracted from its distribution easily. Following this, a question arises, namely under which condition the evaluation cost on P_a can be lower than that on P_b based the simple FOT? It can be expressed as:

$$\begin{aligned}
&E(F,P_a,T) \leq E(F,P_b,T)\\
=&c(x_1) + c(x_2) * (1 - p(x_1 < T)) \leq c(x_2) + c(x_1) * (1 - p(x_2 < T))\\
\Leftrightarrow&c(x_1) * p(x_2 < T) \leq c(x_2) * p(x_1 < T)\\
\Leftrightarrow&\frac{c(x_1)}{p(x_1 < T)} \leq \frac{c(x_2)}{p(x_2 < T)}\\
\Leftrightarrow&cp(x_1,T) \leq cp_(x_2,T)
\end{aligned} \tag{4.2.1}$$

which means, the optimal evaluation plan can be determined by the effective costs of basic evaluations in this case, namely a plan which sort the effective costs of basic evaluations increasingly. Intuitively a smaller effective cost should be carried out earlier, which is quite plausible, since we want to decrease the overall evaluation cost, which requires to check “cheap” evaluation (lower cost) at first generally, on the other hand, the quality should be approved as “soon” as possible (higher probability for approving).

Apart from this simple case, we give an important lemma to obtain an optimal evaluation plan based on the effective costs in a simple FOT.

Lemma 4.10 (Bubble Rule)**.** *Given an FOT F with only one aggregation operator (the minimum t-norm) and n basic evaluations, an evaluation plan P related to F, a bubble rule indicates that if one basic evaluation, say at position k ($1 \leq k < n$), has a higher effective cost than its successor one in P, then a new plan P' by swapping(bubbling) the positions of these both basic evaluations in P will lead to a lower evaluation cost than P for negative approving. Formally:*

$$\exists k \in \{1,\ldots,n-1\} : cp(x_{P_k},T) > cp(x_{P_{k+1}},T) \Rightarrow E(F,P',T) < E(F,P,T)$$

where T is a given threshold, $cp(x_i,T)$ is the effective cost of a basic evaluation x_i for negative approving, and P' is defined by:

$$P'(i) = \begin{cases} P(k+1) & \text{if } i = k \\ P(k) & \text{if } i = k+1 \\ P(i) & \text{otherwise} \end{cases}$$

for all $i \in \{1,\ldots,n\}$.

Proof. Obviously Equation 4.2.1 evidences the bubble rule in case of $n = 2$. To simplify following proof statements, we use c_i to replace $c(x_{P(i)})$ for the evaluation cost of i-th basic evaluation in P, p_i instead of $p(x_{P(i)} < T)$ for the probability of $P(i)$-th basic evaluation having an evaluation degree lower than T. Then we have:

$$\begin{aligned}
E(F,P,T) &= \sum_{i=1}^{n} c(P(i))(1-p(Q_<(i-1,F,P,T)=1)) \\
&= \underbrace{c_1 + c_2(1-p_1) + \cdots + c_{k-1}(1-p_1)\cdots(1-p_{k-2})}_{L} \\
&+ \underbrace{c_k(1-p_1)\cdots(1-p_{k-1}) + c_{k+1}(1-p_1)\cdots(1-p_{k-1})(1-p_k)}_{M} \\
&+ \underbrace{c_{k+2}(1-p_1)\cdots(1-p_{k+1}) + \cdots + c_n(1-p_1)\cdots(1-p_{n-1})}_{R} \\
E(F,P',T) &= \sum_{i=1}^{n} c(P'(i))(1-p(Q_<(i-1,F,P',T)=1)) \\
&= \underbrace{c_1 + c_2(1-p_1) + \cdots + c_{k-1}(1-p_1)\cdots(1-p_{k-2})}_{L} \\
&+ \underbrace{c_{k+1}(1-p_1)\cdots(1-p_{k-1}) + c_k(1-p_1)\cdots(1-p_{k-1})(1-p_{k+1})}_{M} \\
&+ \underbrace{c_{k+2}(1-p_1)\cdots(1-p_{k+1}) + \cdots + c_n(1-p_1)\cdots(1-p_{n-1})}_{R}
\end{aligned}$$

Actually the evaluation cost under P and P' can be divided into three parts, marked in previous form as L, M and R, since the L and R parts under P and P' are identical, we need only to prove that the M part under P, say M_P, is bigger or equal than one under P', say $M_{P'}$. Furthermore we can extend the M part in following way:

$$\begin{aligned}
M_P \le M_{P'} \quad \Leftrightarrow \quad & (1-p_1)\cdots(1-p_{k-1})(c_k + c_{k+1}(1-p_k)) \\
& \le (1-p_1)\cdots(1-p_{k-1})(c_{k+1} + c_k(1-p_{k+1})) \\
\Leftrightarrow \quad & c_k + c_{k+1}(1-p_k) \le c_{k+1} + c_k(1-p_{k+1}) \\
\Leftrightarrow \quad & \frac{c_{k+1}}{p_{k+1}} \le \frac{c_k}{p_k} \\
\Leftrightarrow \quad & cp(x_{P(k+1)}, T) \le cp(x_{P(k)}, T)
\end{aligned}$$

□

The bubble rule provides a method to build an optimal evaluation plan in a special case, namely there is only one aggregation operator (the minimum t-norm) in FOT. In the proof of the bubble rule, a special property of the minimum t-norm is applied, namely the result of the minimum t-norm only depends on one (minimal) basic evaluation, but independent of other ones, which is reflected in computing the evaluation cost as:

$$\begin{aligned}
E(F,P,T) &= c_1 + c_2(1-p_1) + \cdots + c_k(1-p_1)\cdots(1-p_{k-1}) + \ldots \\
&= \sum_{i=1}^{n} c_i \prod_{j=1}^{i-1} (1-p_j) \qquad (4.2.2)
\end{aligned}$$

the cost of k-th basic evaluation would only be taken into account, when the overall quality can not be negatively approved according all carried basic evaluations so far. Regarding this, the bubble rule does not hold for other t-norms generally, for example considering the product t-norm, assume that two basic evaluations are already carried out and both return a degree 0.8, in order to approve the overall quality according to a threshold $T = 0.7$, according to the bubble rule, one has to consider other basic evaluations, since both degrees are greater than T. However from the definition of the product t-norm, one can easily induce that the overall quality can not be over than 0.7, therefore it can be approved negatively without checking further basic evaluations.

Nevertheless, according to the special role of the minimum t-norm (it is the greatest t-norm), the bubble rule estimates the evaluation cost in a pessimistic way as applied on other t-norms, as if it always assumes a basic evaluation as 1 if it turns out that a carried basic evaluation fails approving the overall quality negatively, since 1 is the neutral element for all t-norms. The overall quality is thus assumed only to depend on unknown basic evaluations, but independent of already carried basic evaluations. From this point of view, the bubble rule can be used to draw an upper bound on evaluation cost of a simple FOT for any t-norm. Let us call the evaluation cost in Equation 4.2.2 the *pessimistic cost* for purpose of negative approving in this case.

Actually according the bubble rule, one just need sort all basic evaluations according their effective costs increasingly, their order becomes to an evaluation plan with pessimistic cost.

Definition 4.11 (Ordering)**.** An *ordering*, denoted as O, based on a set of basic evaluations with effective costs is defined as the permutation of indices of these basic evaluations, so that their effective costs are sorted increasingly, formally for a simple FOT with n basic evaluations:

$$O : \{1,\ldots,n\} \to \{1,\ldots,n\}$$

and

$$\forall i,j \in \{1,\ldots,n\} : cp(x_i,T) < cp(x_j,T) \Rightarrow O(i) < O(j)$$

Lemma 4.12 (Upper Bound of Evaluation Cost)**.** *For an simple FOT F with only one t-norm and n basic evaluations with distribution D, and a predefined threshold T, an evaluation plan P can be built by using an ordering on all basic evaluations for purpose of negative approving, then for any optimal evaluation plan P^*, it holds:*

$$E(F,P^*,T) \leq E(F,P,T)$$

Proof. The proof of this lemma can be obtained directly from the boundary rule of t-norm. □

Analog in the case of t-conorm, one can easily deduce the bubble rule on a simple FOT with a t-conorm for positive approving regarding the boundary rule of t-conorm. Furthermore, an upper bound of evaluation cost can be drawn by building an evaluation plan using an ordering for propose of positive approving. In contrast to t-norm and t-conorm, there is no universal boundary rule for the mixnorm, which is strongly dependent on the parameter specification. In case of t-norm and t-conorm, the interval can be tightened in one direction, for example, the interval in case of a t-norm (t-conorm) can be expressed in a general form of $[0,t)$ ($[t,1]$) with $t \geq 0$ in any evaluation step. But in case of a mixnorm, the interval would be tightened in both directions during the evaluation process, for example, assume the aggregation operator A in Figure 4.6 is an OWA mixnorm (with parameters $w_1,\ldots,w_n$), then the interval after the first k basic evaluations is:

$$[F^{(k)}] = [\sum_{i=1}^{k} w_i y_i, \sum_{i=1}^{n-k} w_i + \sum_{i=1}^{k} w_{n-k+i} y_i]$$

where y_i is the i-th largest element in the first k basic evaluations. Namely, the lower bound of $[F^{(k)}]$ is achieved if all remaining basic evaluations return zero, whereas the upper bound of $[F^{(k)}]$ is achieved if all remaining basic evaluations return one. From the previous example, one can see that the interval in case of mixnorm depends on the parameter specification of mixnorm and already known basic evaluations very strongly. Therefore, we do not analyse this case explicitly, but simply use ordering on all basic evaluations to get an evaluation plan. As later the experiments indicate, alone the considerations on t-norm and t-conorm have shown very good performance.

Taking all previous results together, Table 4.2 summaries the considerations based on FOTs in a simple form, where $-$ denotes the case that it is expected that the quality can be only approved after all basic evaluations have been carried out, which means the evaluation cost would be maximal. For instance, for an FOT with a t-norm, the quality can only be approved positively, when all basic evaluations have been carried out. So in all cases marked with $-$, an evaluation plan would be again generated by taking an ordering on all basic evaluations.

Table 4.2: Evaluation plan on FOTs in a simple form

Type of A	Positive approving	Negative approving
t-norm	$-$	Ordering
t-conorm	Ordering	$-$
mix-norm	$-$	

Finally for a simple FOT, one can always generate an evaluation plan using ordering, either for positive and negative approving, which is independent of the type of aggregation operator in the root node. Although according to Table 4.2, the bubble rule only works in two cases under some constraints, we will see later that the resulted evaluation plans can achieve very good performance.

4.2.3 On Hierarchical Fuzzy Operator Trees

As mentioned before, an FOT can be constructed recursively regarding its hierarchical structure. Apart from the previous conclusions on a simple FOT, we can apply the bubble rule in an FOT in a similar recursive manner to generate an overall evaluation plan. The challenge here lies in how to get the cost and distribution for interior nodes, since they are only provided for all basic evaluations. Figure 4.7 illustrates the costs and distributions to be estimated (denoted with $(?,?)$) on the car selection example, where the basic evaluations are assigned with fixed costs c_i and distributions D_i. In this section, so long as no misunderstanding arises, we simply write $c(x_i)$ to c_i and $D(x_i)$ to D_i.

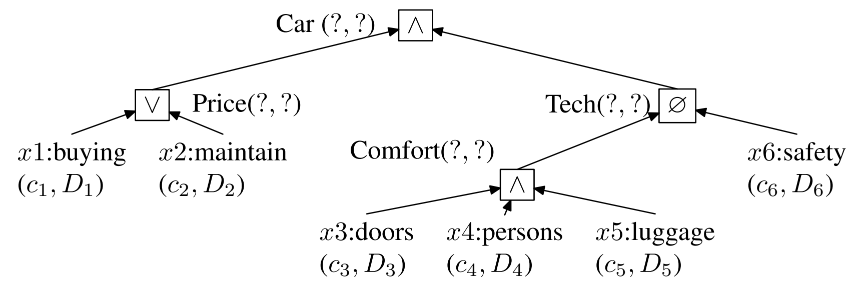


Figure 4.7: Estimating evaluation cost and distribution for interior nodes

It is clear that if cost and distribution for interior nodes can be calculated correctly, the conclusions from previous subsection can be applied in any interior node, in which any its children would be treated as a basic evaluation. We intend to estimate the cost and corresponding distribution for interior nodes recursively in a bottom-up way. Again, let us consider a general case in a two-level-hierarchy (see Figure 4.8), where A denotes an aggregation operator and x stands for an aggregation operator with estimated cost and distribution or a basic evaluation with given cost and distribution. Obviously an optimal evaluation plan might be in this simple case an ordering on all basic evaluations, denotes as P, according to a given threshold T. The evaluation plan here can be combined recursively also, that is

$$P := [P_{x_1}, P_{x_2}, \dots, P_{x_n}]$$

if it holds $cp(x_1,T) \leq cp(x_2,T) \leq \dots \leq cp(x_n,T)$ regarding to T, where P_{x_i} is the evaluation plan on x_i in turn. Namely the evaluation plan on A is a sequential combination of the evaluation plan on basic evaluations, which are sorted by its (estimated) effective costs.

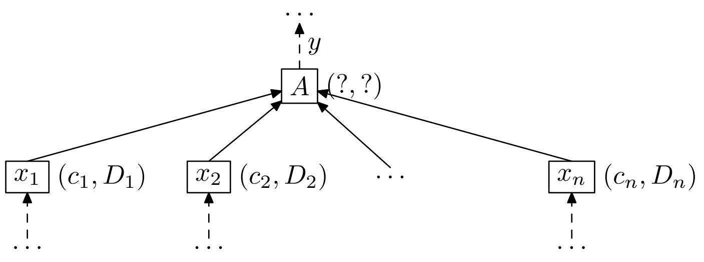


Figure 4.8: Estimation on evaluation cost and distribution

As long as P is determined, we can estimate the (upper bound of) evaluation cost of A regarding the type of A:

- For purpose of negative approving, if A is a t-norm, then:

$$\begin{aligned} c(A) &:= c(1) + c(2)(1-p(1)) + \dots \\ &\quad + \dots + c(n)(1-p(1))(1-p(2))\dots(1-p(n-1)) \\ &:= \sum_{i=1}^{n} c(i) \prod_{j=1}^{i-1} (1-p(j)) \end{aligned} \tag{4.2.3}$$

  This also holds in the case of positive approving and A is a t-conorm;

- Otherwise, the evaluation cost of A is considered in the worst case, namely the sum of all costs of its children nodes, then:

$$c(A) := \sum_{i=1}^{n} c(i)$$

where $c(i)$ is the evaluation cost on $x_{P(i)}$, $p(i)$ denotes $p(x_{P(i)} < T)$ in case of negative approving and $p(x_{P(i)} \geq T)$ in case of positive approving.

The distribution of A's output, say D_y, can also be estimated based on the known distributions on $x_1, \dots, x_n$, which is denoted as *joint distribution*. Formally we have the probability density function of D_y:

$$\begin{array}{ll} \text{Let} & D(a_1, \dots, a_n) = \prod_{i=1}^{n} p(x_i = a_i) \\ & \text{with } x_i \sim D_i \\ \text{Then} & y = A(x_1, \dots, x_n) \\ & D_y \text{ is the image of } D \text{ under } A. \end{array} \tag{4.2.4}$$

under the assumption that $x_1, \dots, x_n$ are independent of each other.

Theoretically, a joint distribution of aggregation operator can be determined precisely, if the distributions of children nodes are known. However from the technical point of view, it is impossible to calculate D_y in Equation 4.2.4 pointwise. Instead, we try to estimate the joint distribution in an interval manner. A naive method to estimate a joint distribution is developed in this section, which is illustrated in Figure 4.9, where a joint distribution of $z = f(x,y)$ based on both distribution x and y is estimated in an interval manner, f is an aggregation function on x and y. For a given integer number q, the distributions of x and y are divided into q equi-width intervals. Each pair of intervals from x and y builds a representative point for the joint distribution of z, whose probability is the product of probabilities reading from x and y, in this example

$$p_z = p_x \cdot p_y = 0.2 \cdot 0.05 = 0.01$$

and has a value of $z = f(x,y)$, in this example

$$z = f(x,y) = f(0.7, 0.3)$$

by taking the middle points of intervals on x and y. At the end, a histogram on all these representative points estimates the joint distribution on z. In principle, as q goes to infinity where the interval is divided small enough, the joint distribution can be estimated precisely.

4.2.4 A Static Algorithm

Now we can give the first algorithm to estimate an optimal evaluation plan, as Algorithm 4.1 shows. After recursively estimating the distribution and evaluation cost for

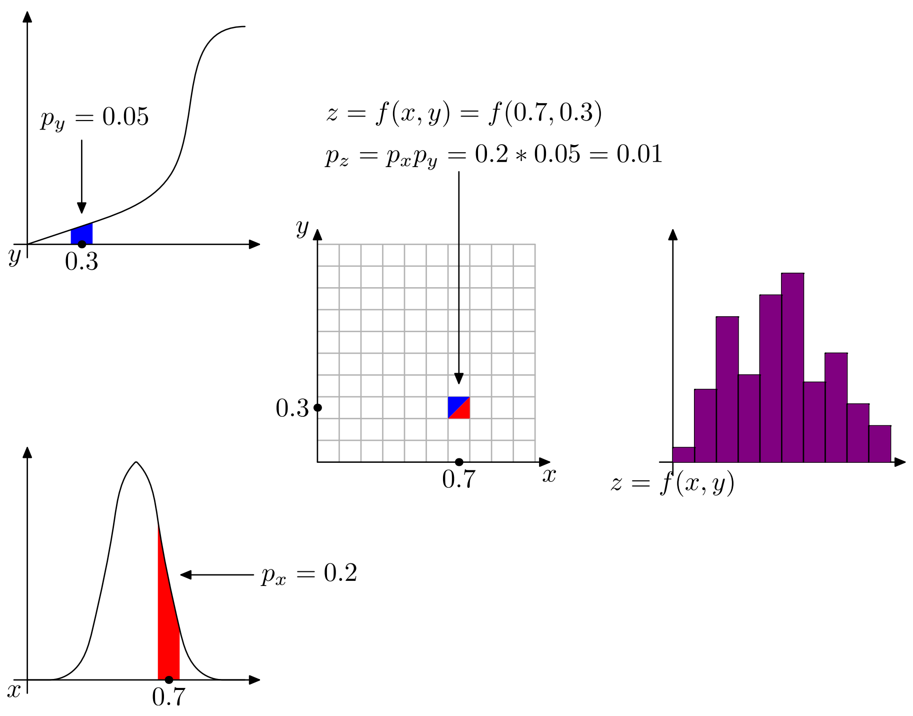

$p_y = 0.05$
y
0.3
$z = f(x,y) = f(0.7, 0.3)$
$p_z = p_x p_y = 0.2 * 0.05 = 0.01$
y
0.3
0.7
x
$z = f(x,y)$
$p_x = 0.2$
x
0.7

Figure 4.9: Joint distribution

all interior nodes in both positive and negative approving cases, the lower estimated evaluation cost is selected to generate an evaluation plan in the line 6.

Algorithm 4.1: A static algorithm to determine evaluation plan

```
   Input: FOT $F$, distribution $D$, threshold $T$
   Output: Evaluation plan $P$
 1 begin
 2     estimate joint distributions for all interior nodes in $F$ ;
 3     $r \leftarrow$ root of $F$;
 4     $c_{pos} \leftarrow$ estimateCost ($F$, $r$, positive) ;
 5     $c_{neg} \leftarrow$ estimateCost ($F$, $r$, negative) ;
 6     if ($c_{neg} \geq c_{pos}$) then
 7         $P \leftarrow$ getEstimatePlan($F$, $r$, positive) ;
 8     else
 9         $P \leftarrow$ getEstimatePlan($F$, $r$, negative) ;
10     end
11 end
```

In order to illustrate the idea in Algorithm 4.1, Figure 4.10 demonstrates how to combine an evaluation plan recursively on the car selection example, where the (combined) evaluation plans are listed for interior nodes and root node. The $P : [3,5,4,6]$ at the right side of "Tech" gives an order on four basic evaluations involved in the subtree on "Tech", which is in turn combined into the plan on its parent node "Car".

As mentioned before, a static algorithm builds an evaluation plan by using an ordering based on the effective costs of all basic evaluations, the resulted evaluation plan (referred as *static plan* in following text) delivers an evaluation cost in a pessimistic case when applied to evaluate data. The evaluation cost of static plan indicates an upper bound of evaluation cost of an optimal plan.

However a static plan lacks of flexibility, in which a static plan shows the *partial completeness* regarding to basic evaluations from subtrees.

Definition 4.13 (Partial Completeness)**.** Given an FOT F, an evaluation plan P based

```
Function estimateCost(F,r,p)
    Input: FOT $F$, Node $r$, Is positive approving? $p$
    Output: Evaluation cost at $r$
1   begin
        // estimate evaluation costs of non-leaf children
2       for ($n \in \{$ non-leaf children of $r\}$) do
3           $cost_n \leftarrow$ estimateCost($F$, $n$, $p$)
4       end
5       $cost \leftarrow 0$ ;
6       if ($r$ is a t-norm $\wedge$ $p$ is false) then
            // using Ordering for negative approving on t-norm
7           $cost \leftarrow$ apply Equation 4.2.3 ;
8       else if ($r$ is a t-conorm $\wedge$ $p$ is true) then
            // using Ordering for positive approving on t-conorm
9           $cost \leftarrow$ apply Equation 4.2.3 ;
10      else
            // sum of all evaluation costs
11          for ($n \in \{$ children of $r\}$) do $cost \leftarrow cost + cost_n$;
12      end
13  end
14  return cost
```

```
Function getEvaluationPlan(F,r,p)
    Input: FOT $F$, Node $r$, Is positive approving? $p$
    Output: Evaluation plan at $r$
1   begin
        // get evaluation plan of non-leaf children
2       for ($n \in \{$non-leaf children of $r\}$) do
3           $P_n \leftarrow$ getEvaluationPlan($F$, $n$, $p$)
4       end
        // take ordering as evaluation plan on all children nodes
5       $P \leftarrow$ combine all $P_n$ after their (estimated) effective costs ;
6   end
7   return P
```

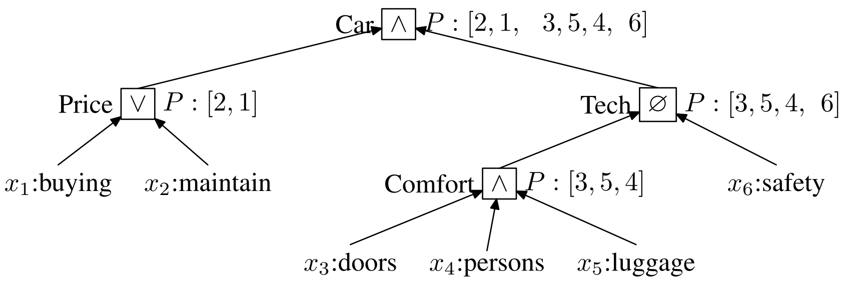


Figure 4.10: Combine evaluation plan recursively

on F, P is called *partial complete*, if for any two nodes in F, say p_1 and p_2, which have common parent, all basic evaluations involved in p_1 must be carried out either before or after those basic evaluations involved in p_2.

Because an FOT can be constructed in a recursive manner, so any two nodes in an FOT with common parent can be viewed as two FOTs (subtrees) again, which can be either a basic evaluation, or an FOT in turn. Since a static plan of parent node is sequential combination of static plans from children nodes, according to a static plan, a subtree (all its involved basic evaluations) must be evaluated either before or after another subtree completely, a plan containing basic evaluations from different subtrees alternatively is not allowed. For example, on the car selection problem we have:

$$\begin{aligned}
\text{Allowed plans:} & [1,2,\quad 3,4,5,\quad 6] \\
& [4,5,3,\quad 6,\quad 1,2] \\
& [6,\quad 5,3,4,\quad 2,1] \\
\text{Not allowed plan:} & [2,6,3,1,5,4]
\end{aligned}$$

One can see that the pair of 1 and 2 (“Buying” and “Maintain”) from a subtree of “Price” appears always together in allowed plans, as well as the pair of $3,4$ and 5.

The partial completeness of a static plan causes to carry out some superfluous basic evaluations, even though they can not contribute to approve the overall quality. To illustrate this problem clearly, let us reconsider the car selection problem in a similar situation as Figure 4.4 shows, namely according a static plan, the basic evaluations would be carried out one by one beginning from “Buying” rightwards ($P_1 = [1,2,3,4,5,6]$) with a given threshold $T = 0.8$. Figure 4.11 demonstrates this example in detail. In order to approve the overall quality positively, let us assume that the basic evaluation on “Buying” turns out 0.8, from which the interval of “Price” and “Car” can be tightened due to the boundary property of t-norm and t-conorm respectively. According to the static plan P_1, the basic evaluation “Maintain” should be carried out. However the contribution of evaluating “Maintain” is very low on the overall quality of “Car”, the output of “Maintain” can tighten the interval of “Price” at the most, but have no influence of the interval of “Car”. In this case, evaluating “Maintain" can be considered as superfluous, so any plan that prefers to select other basic evaluations except “Maintain” might require less evaluation costs than the static plan P_1.

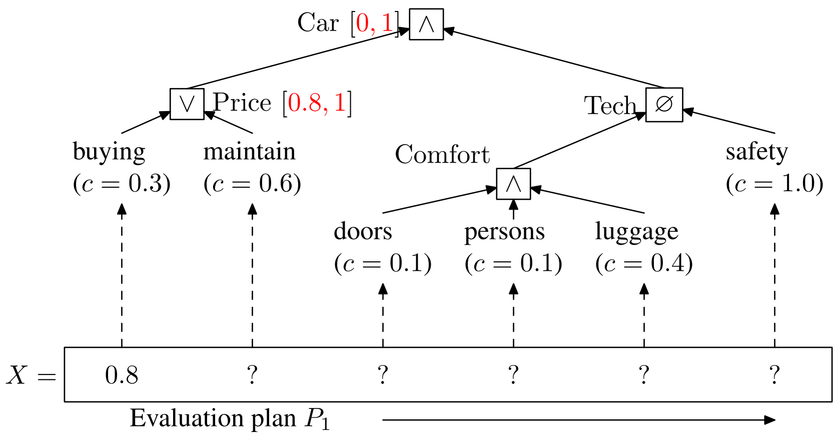


Figure 4.11: Partial completeness of static plan

Due to the partial completeness, the evaluation cost of a static plan is usually higher than it really expected for quality approving. Generally speaking, how the partial completeness affects the overall evaluation cost depends on the complexity of an

FOT, since the more complicate an FOT is, in terms of number of involved aggregation operators, the more superfluous evaluation costs might be required. In a extreme simple case, where an FOT has only one aggregation operator, its effect of evaluation cost disappears.

Though due to the partial completeness, a static plan is expected to carry out some superfluous basic evaluations, a static plan is still very useful in practice. For instance, it can be used to provide a general impression for an optimal evaluation plan before handling any concrete data. Generally, a static plan can indicate the importance of different basic evaluations for quality approving. Furthermore, as we will later see, a static plan still outperforms other plans in most cases, which are generated in a greedy or random manner.

4.2.5 An Online Algorithm

An *online algorithm* is developed here to decrease the influence of the partial completeness in a static plan. Taking a closer look at the partial completeness of a static plan, one can easily find out that the problem lies in how to detect the superfluous basic evaluations effectively, in other words, how to select a basic evaluation to carry out, which has the most contribution in purpose of quality approving.

The idea behind the online algorithm is that instead of deducing a complete evaluation plan at one time, we try to determine one step each time in an evaluation plan, namely only one basic evaluation is selected and carried out, which has most contribution for approving the overall quality. In order to determine such a basic evaluation, a pseudo-plan is generated just as the static algorithm does, the first basic evaluation indicated by this plan is selected in the online algorithm and carried out next. Note that a static plan gives an order of basic evaluations, which can be viewed as an order of importance to contribute for quality approving, the earlier does a basic evaluation appears in the order, the more contribution it is expected. So taking the head of a static plan to carry out is plausible in the online algorithm to achieve the “most” contribution for quality approving.

Afterward it is possible to update the estimated evaluation costs and distributions on all interior nodes according to newly carried basic evaluation. On which a new static plan can be generated according to updated evaluation costs and distributions, then the next basic evaluation can be determined recursively. Being applied to evaluate an object, an online algorithm tries to update the estimations on interior nodes every time after a basic evaluation is carried out. Algorithm 4.4 gives a pseudo-code for this idea, where so long as the overall quality is not approved, the algorithm enters a loop. In the loop, the same workflow is called as in the static algorithm, the only difference lies in that just one basic evaluation determined by either positive or negative approving is selected in the line 10. Finally the selected basic evaluation is carried out in the line 20, and its evaluation will be taken into account in next iteration for further estimation and updating.

Notice that an online algorithm avoids the effect of the partial completeness in a static plan, in which every time a new static plan would be generated based on updated evaluation costs and distributions. If some basic evaluations are turned out that they have little influence for purpose of quality approving, they will appear in the middle or back part of the new static plan, so they are not selected in the online algorithm. As a result, they do not have to be evaluated as the partial completeness requires, the overall evaluation cost would be kept as lower as possible.

Figure 4.12 demonstrates the third iteration by applying this algorithm on the car selection example, where the head of evaluation plan ($P = [2,3]$) is determined by

Algorithm 4.4: An online algorithm to determine evaluation plan

```
    Input: FOT $F$, distribution $D$, object $o$, threshold $T$
    Output: Evaluation plan $P$
1   begin
2       $index \leftarrow 1$ ;
3       $n \leftarrow$ number of basic evaluations in $F$ ;
4       selected $\leftarrow \emptyset$ ;
5       while (Quality of o is not approved) do
6           update joint distributions for all interior nodes in $F$ ;
7           $r \leftarrow$ root of $F$;
8           $c_{pos} \leftarrow$ estimateCost ($F$, $r$, positive) ;
9           $c_{neg} \leftarrow$ estimateCost ($F$, $r$, negative) ;
10          if ($c_{neg} \geq c_{pos}$) then
11              $P_{pos} \leftarrow$ getEstimatePlan ($F$, $r$, positive) ;
12              $i \leftarrow P_{pos}(1)$ ;
13          else
14              $P_{neg} \leftarrow$ getEstimatePlan ($F$, $r$, negative) ;
15              $i \leftarrow P_{neg}(1)$ ;
16          end
17          $P(index) \leftarrow i$ ;
18          selected $\leftarrow$ selected $\cup i$;
19          $index \leftarrow index + 1$;
20          carry out the $i$-th basic evaluation ;
21      end
        // insert remaining basic evaluation into $P$
22      for (each $i \in \{1, \ldots, n\} \setminus$ selected) do
23          $P(index) \leftarrow i$ ;
24          $index \leftarrow index + 1$
25      end
26      return $P$
27  end
```

first two iterations, and the third iteration aims to determine the index (i) of basic evaluation to be carried out next. As "Maintain" and "Doors" are already evaluated, their evaluations (0.5 and 0.8) are considered to update the estimated evaluation costs and distributions for interior nodes. The optimal evaluation plan on "Comfort" is denoted as $P : [5, \ldots]$, because only the first element is interested in this step. To this end, the evaluation plan P is extended with 5, namely "Luggage" should be evaluated as next.

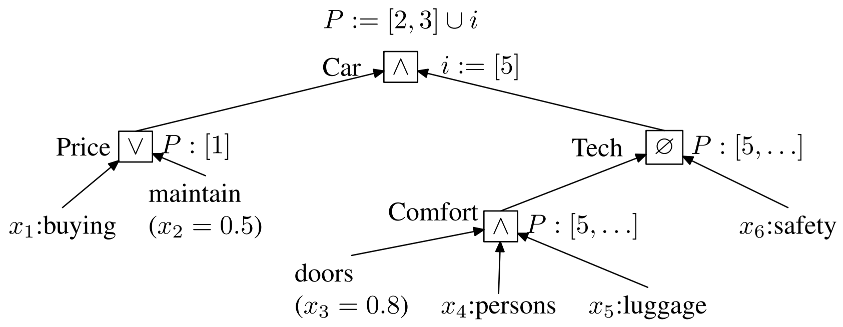


Figure 4.12: Online algorithm to determine an evaluation plan

4.2.6 On Correlated Data

An important requirement by estimating joint distribution is the independence among children nodes (see Equation 4.2.4), it might become difficulties to be fulfilled in practice, for instance if the "Safety" factor is positively correlated with "Buying" in the car selection problem, namely the safer a car is, the higher purchase price would be expected, and vice verse.

In the case of correlated data, the joint distribution can only be deduced using the original equation 4.2.4, in which the conditional probability is required. Consequently the naive method to estimate a joint distribution described in Figure 4.9 does not work any more. We give a sophisticated method on correlated data, in the case that the analysis of correlation is not trivial or even impossible.

Since our goal here is to find out the distribution on interior nodes in an FOT of intermediate evaluations, we try to estimate the distribution directly instead from its probabilistic density functions, in which every basic evaluation is treated as a random number following a known distribution, and every complete evaluation process on FOT leaves an intermediate output for each interior node. Theoretically if the evaluation process can be rearranged in a random manner and be carried out enough many times, the outputs of an interior node can be used to estimate its distribution, for example using a histogram. Remember the training data provided for calibration of FOTs contains exemplary data, which can be used to estimate the distribution on interior nodes. Notice that this method can be applied on independent data also, as long as there are sufficient training data available.

Unfortunately this method has its limitation that it only works in the case with sufficient training data. If only insufficient training data is provided, the estimated distribution might not be representative.

4.2.7 Complexity Analysis

Assume that the evaluation cost and distribution of basic evaluations can be assessed in constant time, the procedure to compute the Ordering has the complexity of $O(n^2)$ in worst case by using sorting algorithm, where n is the number of basic evaluations involved in an FOT. However, the most time-consuming part for two algorithms introduced in this section is to estimate the joint distribution of interior nodes. According to the idea of estimating the joint distribution, generating a static plan has a time complexity $O(c \cdot q^k)$ with c the complexity of an FOT and q the number of representative points of distribution, k is the maximal number of children of an aggregation operator. The time complexity of generating an online plan depends on the number of really carried basic evaluations and their positions in an FOT, generally in average case it can be written as: $O(mc \cdot q^k)$ with m the expected number of really carried basic evaluations. In the worst case, where all basic evaluation have to carried out, the time complexity of generating an online plan is $O(nc \cdot q^k)$.

4.3 Experimental Evaluation

In this section the performance of evaluation cost minimization techniques on FOTs is investigated. The experiment setup is introduced in Section 4.3.1, before the experiments and results are shown in Section 4.3.2.

4.3.1 Experiment Setup

To make a comparison, we construct two evaluation plans in a random and greedy manner respectively, which are used to compare the evaluation plan generated by the static or online algorithms introduced before. These two plans are defined as follows:

- **Random plan** : A random plan is generated as a random permutation upon the indices of basic evaluations.
- **Greedy plan** : A greedy plan is an order of indices of basic evaluations sorted by their evaluation costs ascendingly. A greedy plan intends to select the "cheapest" basic evaluation at each step, so that the evaluation cost can be kept as lower as possible, but it ignores the distributions of basic evaluations completely.

For experiments in this section, FOTs are created randomly as in the experimental section 3.4. There are two main changes made: firstly, the number of children of an aggregation operator is chosen randomly from $\{2,3,4\}$ instead of $\{2\}$, that is, we allow an aggregation operator to have two until four children randomly, since this can reflect the influence of type of aggregation operators, as Section 4.2.2 mentioned. The second change lies in that different from creating complete FOTs by fulfilling all leaf nodes with randomly generated fuzzy sets, the FOTs involved in this experiment are fulfilled directly with numeric variables in the unit interval $[0,1]$, which ought to simulate the basic evaluations returned by fuzzy sets. Moreover every input variable is assigned with a random evaluation cost and distribution. The former is simply a numeric value normalized into a unit interval $[0,1]$ following the uniform distribution. To generate a random distribution of inputs of an FOT, a fixed number of representative points are generated, whose height (randomly selected in the unit interval $[0,2]$) denotes a probability density for a fixed x value, then we simulate a continuous probability density function in $[0,1]$ by connecting the representative points, as Figure 4.13 demonstrates. To make sure that the area under the probability density function is 1, which indicates the sum of probabilities in the interval $[0,1]$, these representative points can be adjusted proportionally in order to make the under area equal 1 approximately. The adjusted probability density function is plotted with solid line in Figure 4.13.

Under a random distribution, the output of a basic evaluation can be viewed as a random variable with a fixed probability density function. To generate random outputs of a basic evaluation in this case, the *cumulative distribution function* (CDF) is

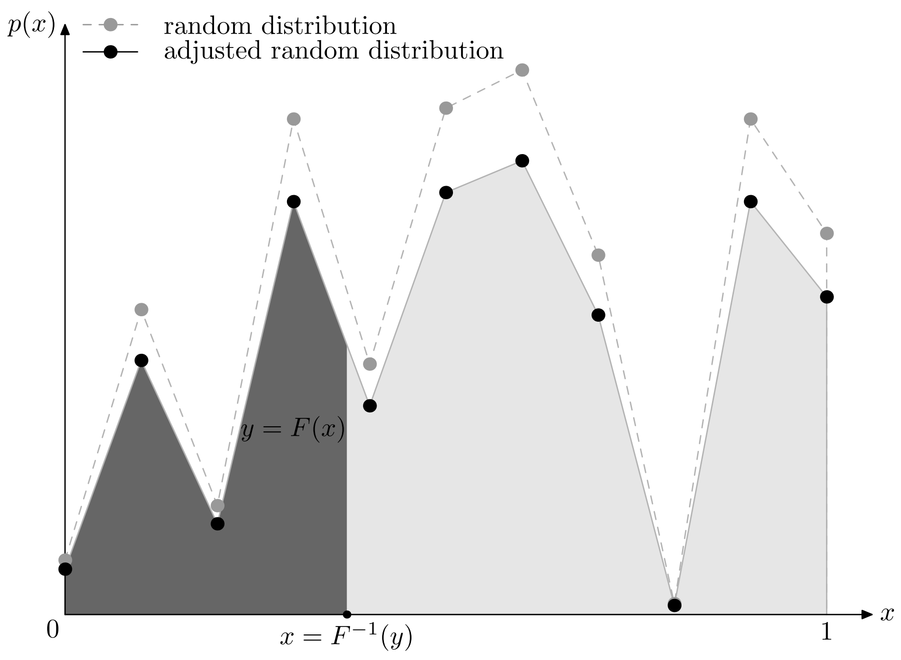


Figure 4.13: Generating random distribution and random number

applied here, which is defined as:

$$F(x) = \int_{-\infty}^{x} p(x)dx$$

with $p(x)$ the probability density function. At first, a random number in the unit interval $[0,1]$ is generated under the uniform distribution, denoted as y here. Then we intend to find an x, which cause to $F(x) = y$ as Figure 4.13 indicates, this can be obtained by using the inverse function of the CDF, namely $x = F^{-1}(y)$.

Definition 4.14 (Evaluation Gain)**.** Given an FOT $F(x_1,\ldots,x_n)$, a threshold T, an evaluation plan P based on F, a set of objects $\mathbb{D}$ to be evaluated, the *evaluation gain*, denoted as $EG(F,P,T)$, is defined as the saved evaluation cost of $\mathbb{D}$ under P, which is normalized into the unit interval $[0,1]$. Formally:

$$EG(F,P,T) := 1 - \frac{\sum\limits_{o\in\mathbb{D}} E(o,F,P,T)}{|\mathbb{D}| \cdot E_{full}}$$

where E_{full} is the sum of costs of all involved basic evaluations, namely $E_{full} = \sum\limits_{i=1}^{n} c(x_i)$, and $E(o,F,P,T)$ is required evaluation cost for approving the quality of object o, which is defined by:

$$E(o,F,P,T) := \sum_{i=1}^{n} c(x_{P(i)})(1 - Q(i-1,F,P,T))$$

To evaluate the experimental results, the evaluation gain assigns a number to indicate how much evaluation cost are saved using a plan upon a set of objects. So the nearer an evaluation gain lies to 1, the less evaluation cost is required, and the better the corresponding plan is.

4.3.2 Experimental Results

Let us denote an evaluation plan deduced by the static algorithm as *static plan*, and by the online algorithm as *online plan* respectively. Based on a randomly generated FOT, a fixed number of inputs are produced according to randomly generated distributions, then evaluation cost under static plan, online plan, as well as random and greedy plans are calculated according to a randomly generated threshold. For random plan, a fixed number of random plans is generated to get a stable performance, to this end, the evaluation cost on random plan is the averaged value among a fixed number of random plans (here 100). In this section, following experimental results are averaged among 50 repetitions.

At first, we investigate the influence of complexity of FOTs on evaluation costs, in which the experimental results are grouped by the complexity of FOTs and sorted by the provided mount of data. Figure 4.14 shows the averaged evaluation gains for four evaluation plans (in four representative cases), where the standard deviation is omitted for the sake of clearness.

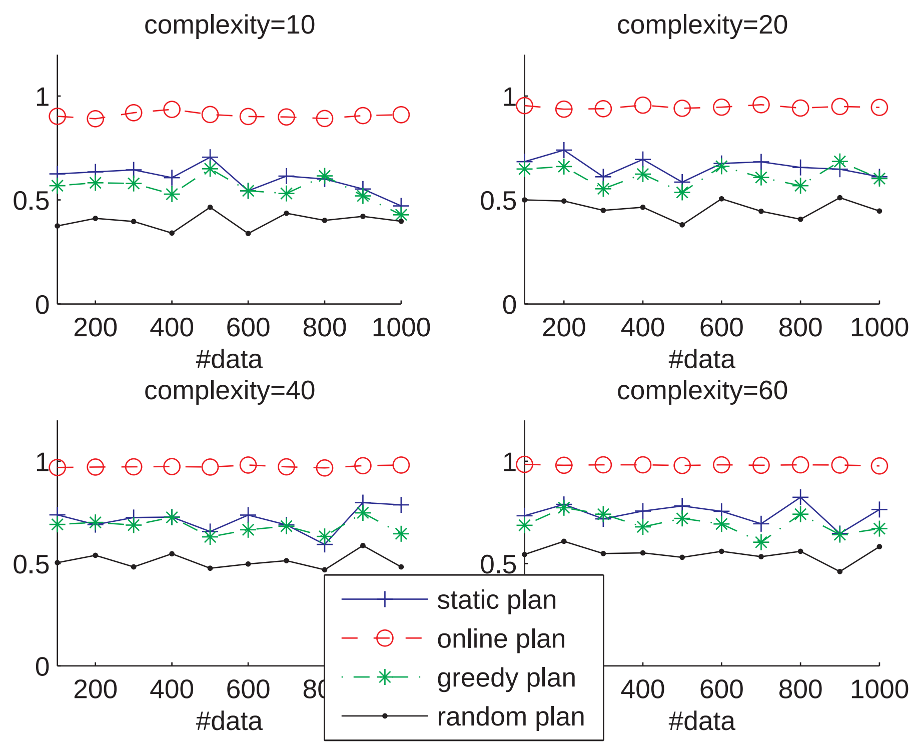


Figure 4.14: Evaluation gains regarding complexity of FOTs

The experimental result is quite promising, an online plan achieves an evaluation gain around 94%(0.958), which corresponds to only 4% of the full evaluation costs for quality approving on the average, and therefore outperforms the other three plans (static plan 68%, greedy plan 64% and random plan 48%) considerably. On the other hand, a static plan generated by static algorithm requires less evaluation cost than a greedy plan in most case (54 in 60 observations), whereas a greedy plan needs generally less evaluation cost than a random one.

As the amount of provided data increases, the performance of considered evaluation plans does not change too much, which can be observed in that the relative position of four evaluation plans is kept with various complexity and amount of data. Interesting is to compare a greedy plan and static plan, theoretically a static plan is expected to require much less evaluation costs than a greedy plan, as more data is provided, since the main difference of both plans is whether they take the distribution of the data into account. As more data is provided, the influence of distribution of data becomes clearer, the superiority of static plan should be easier to be observed in principle. However, the negative effect of partial completeness caused by a static plan becomes clearer at the same time, in which the accumulative costs on superfluous basic evaluations becomes larger, so there is no explicit trend of the superiority of static plan against greedy plan.

Following, the experimental results are reorganized in a reverse order, which is grouped by the amount of provided data and sorted by complexity of FOTs, as Figure 4.15 shows (only in four representative cases here to save place). The similar observations can be drawn from Figure 4.15, an interesting aspect in these experimental results is that the evaluation gains increase as the complexity of FOTs becomes large, no matter which evaluation plan is applied here. Since the number of basic evaluations depends on the complexity of FOTs, the relationship between complexity of FOTs and evaluation cost becomes clear that the more complicate an FOT is, in terms of number of nodes, the less evaluation cost is required generally.

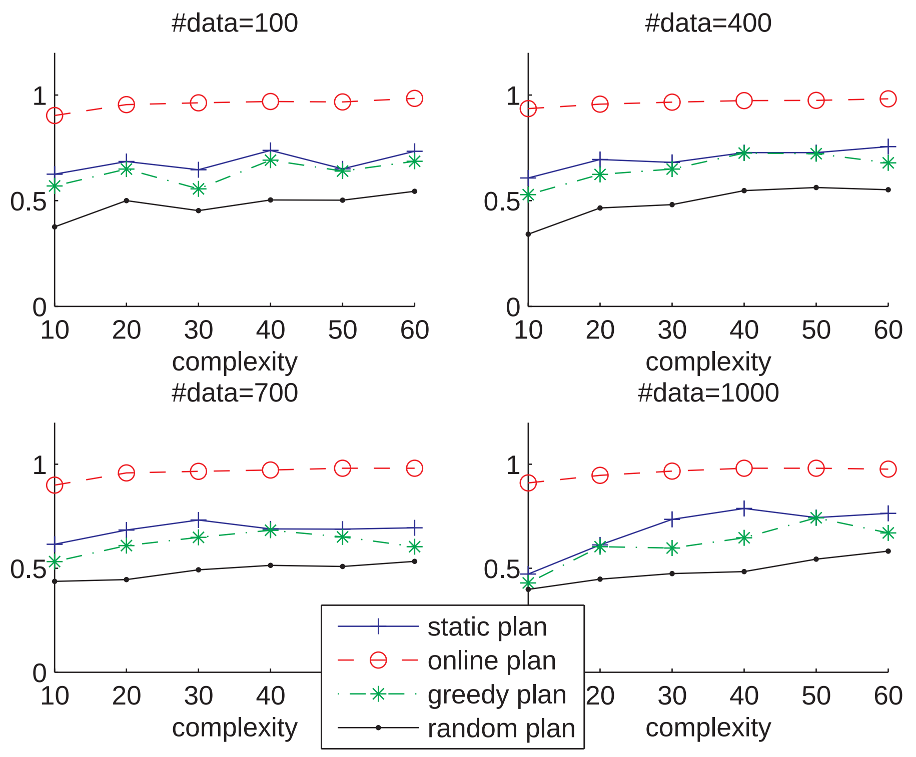


Figure 4.15: Evaluation gains regarding mount of provided data

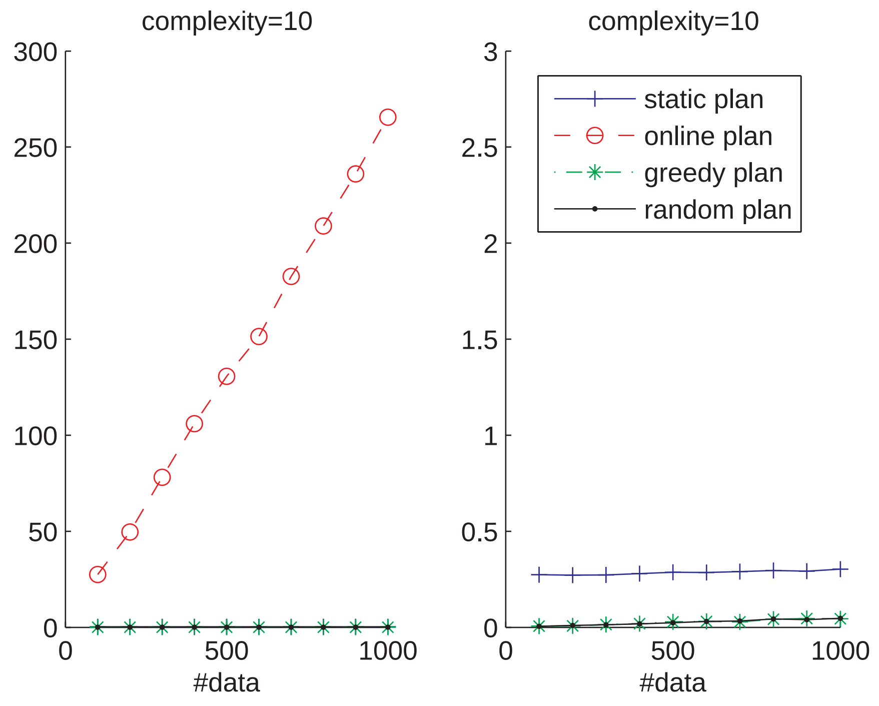
Considering two comparable evaluation plans, a greedy plan and static plan, one can see the clear superiority of static plan on FOTs with smaller complexities particularly, for example with complexity 10 and 20, the evaluation costs of static plan lie clearly under that of greedy plan. But as the complexity of FOTs becomes large, this superiority reduces and vanishes, even in several cases, a greedy plan outperforms a static one. Remember the partial completeness of static plan is caused by the hierarchical structure of FOT, for a simple FOT with only one aggregation operator, the partial completeness of static plan disappears. So the complexity of FOT determines the effect of partial completeness of static plan directly. For FOTs with big complexity, a static plan tends to require more evaluation costs than a greedy plan.

Amongst other, the evaluation cost required by an online plan is quite small, which has also its price that an online algorithm needs much more runtime than other three plans. Figure 4.16[1] shows the runtime used by four evaluation plans representatively (with fixed complexity 10, but different provided data), where an online algorithm has a much higher magnitude of required runtime. Figure 4.17 shows the runtime used by four evaluation plans with 400 data under different complexity. Nevertheless, an

[1] To observe this performance clearly, the runtime of other three plans is plotted in the right half with much smaller magnitude additionally.

online algorithm is still applicable in practice, even for relatively complicated FOTs and big datasets (for example, runtime = 12.4 minute for complexity = 60 and #data = 400).

Figure 4.16: Runtime (s) of four evaluation plans with fixed complexity



Notice that an online algorithm delivers for each to be evaluated object an individual evaluation plan, while a static algorithm predicts only one plan, which ought be able to minimize overall evaluation cost. Regarding this, one might apply these two algorithms in different situations, such as a static algorithm can be used to estimate an optimal evaluation plan before handling any concrete data. On the other hand, an online algorithm provides a dynamic mechanism in order to minimize evaluation cost on concrete data.

4.4 Conclusions

In this chapter we addressed a cost issue on FOTs, since a precise evaluation is not always necessary when applying an FOT model, secondly the basic evaluations of an FOT have different evaluation costs. To estimate an optimal evaluation plan, which ought to require evaluation costs as less as possible, two algorithms are developed in this chapter to estimate optimal evaluation plans. The experimental results show that evaluation plans determined by both algorithms requires significant less evaluation costs than other evaluation plans generated in a random or greedy manner.

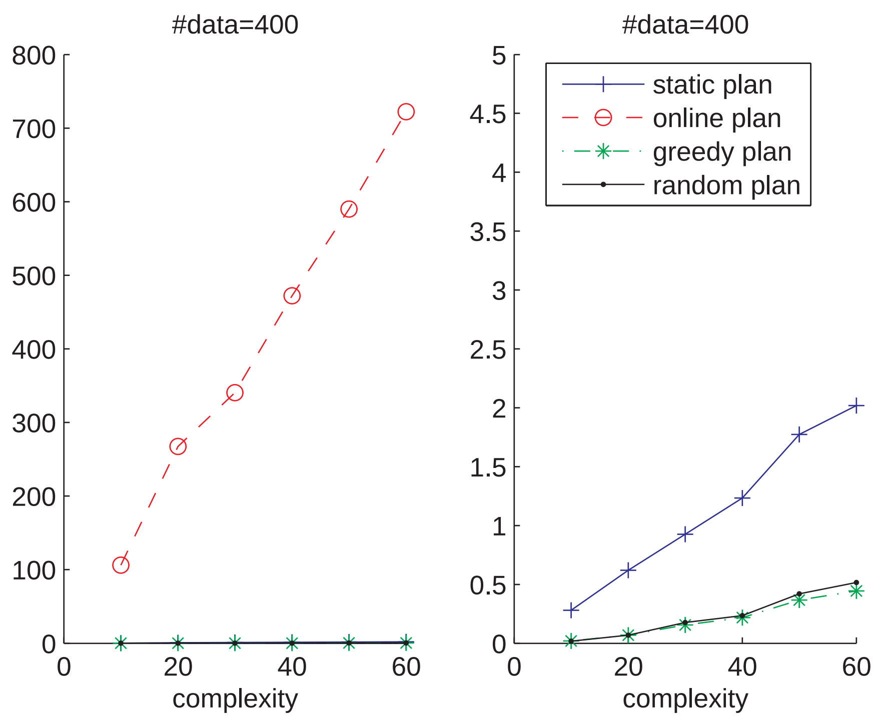


Figure 4.17: Runtime (s) *of four evaluation plans with* 400 *data*

5 Related Work

In this chapter we give an overview of approaches related to the main topics presented in this thesis. As mentioned before, an FOT model first of all serves as a convenient tool for modeling utility functions, with which user preferences and utilities can be modeled. A large number of techniques for modeling utility function have been proposed in last decades. Modeling an FOT is characterized by its tree-like hierarchical structure and the use of fuzzy logic-based operators as nodes. Both techniques are of course not new and there are already a vast number of related works available. Finally, the calibration of FOTs with sample data can be categorized into supervised learning, a well-known machine learning field that has gained increasingly attentions in recent years. As an exhaustive discussion of all related techniques from former aspects is beyond the scope of this work, we primarily concentrate on those techniques that touch on one of the topics of this work in terms of comparable objectives or applied methods. However, we also attempt to provide relevant sources for further investigations in related topics and to point out the main differences to the mentioned approaches. According to the similarity to our works, the related approaches are categorized into the following four classes, that is,

- the idea of utility modeling,
- the idea of hierarchical modeling,
- the use of fuzzy logic-based operators in decision making,
- the calibration of a network-like structure with given input/output data.

Note that this is a very ambiguous classification in the sense that many of them are involved not only in single class, but in several ones, as we shall indicate later.

5.1 Utility Theory and Utility Modeling

The concept of utility was first applied in economics as a measure of the relative satisfaction or desiredness from consumption of goods, which can be used to explain economic behavior and furthermore tries to increase one's utility. The central result of utility theory is a representation theorem that identifies a set of conditions guaranteeing the existence of a function consistent with the preference of a decision maker, the real-valued function is called utility function [NM53]. An overview of the main issues and developments of utility theory in the economic area is given in

[BHS98, BHS04]. Utility theory has been shown to be an efficient tool to help users by decision making, especially under risk and uncertainty, the principle of maximizing expected utility has long been established as the guide to making rational decisions [Sav72, LR57]. In [Cam95], the author summarizes the main contributions in this special topic and provides a comprehensive overview of recent researches. Another survey of the main issues using utility theory for decision making in economics is made by Fishburn in [Fis69].

To model a utility function for a decision problem, it is necessary to elicit a utility model separately for each user, since every individual's utility may be different. Therefore, utility elicitation has been studied extensively in the area of decision analysis (DA) [LR57, How68, How77, KR76]. It has started to gain more attention in medical informatics [FS92, HHB92, HHB95] and artificial intelligence [HHR$^+$03, HH97, CP04, LHL97] recently. General speaking, there are two common approaches to utility function elicitation: The first is to base the determination of the user's utility function solely on elicitation of qualitative preferences, the second makes assumptions about the form and decomposability of the utility function. Decomposable utility functions support more inference and are easier to elicit from user, our approach belongs to this category. In [CP04], the author provides a summary of different techniques for preference elicitation, where the user's preference is represented in the form of utility function (also called value function). Here we just review several of them briefly:

- **Value function elicitation** : In [KR76], the author provides a relatively easy, straightforward framework of eliciting an additive independent value function by creating scales for each component of the value function and querying the user about the behavior of each sub-value function. Under the assumption of additive independence, the (sub-) value functions and scale factors are elicited by asking the user several questions.

- **Preference elicitation by criteria decomposition** : The perhaps most related to our works, a well-known approach is the Analytical Hierarchy Process (AHP), a decision support tool to solve multi-criteria decision problems [Saa94]. Given a set of alternatives and related criteria, it constructs a multi-level hierarchical structure, like our approach, by recursively splitting the criteria into sub-criteria. Then the relative importance (weight) of the decision criteria is obtained in the form of Eigenvalues of matrices, which are elicited from user by pairwise comparisons. The users are proposed to express their preference degree in predefined nine levels to make a comparison between two criteria or measurements, questions like "How much better is A_i than A_j on a criterion?". Nevertheless, difficulties related to AHP have been discussed in several works, for example in [Dye90].

- **Preference elicitation via theory refinement** : A neural-network-like system, Knowledge-Based Artificial Neural Network(KBANN), is proposed in [HHR$^+$03], which starts with approximate and incomplete domain knowledge and then corrects for inaccuracies and incompleteness by training on examples.

- **Case-based preference elicitation** : In [HH03], a case-based preference elicitation model is built by using a new measure to compute the similarity of personal preference, the key advantage of this measure is its extensibility to accommodate partial preferences and uncertainty.

- **Interactive preference elicitation** : In [HS02], the author follows an information theoretic approach to elicit utility functions automatically using user feedback. This approach has been extended and applied to the matchmaking , which offers a general adaptation solution to learn the user preferences with implicit and explicit user feedbacks [FM03a].

Another comprehensive overview of eliciting knowledge from experts is given in [Hof89]. A summary of developments in the field of multi-criteria decision making with the help of utility functions can be found in [PSZ95].

Our work in this thesis can be seen as another approach for the purpose of utility elicitations. However, it differs from these approaches, amongst other, in that it is characterized by its hierarchical structure, which makes use of tools and techniques from fuzzy set theory. Moreover, the calibration of the structure is completely different, as different techniques are applied in previous mentioned approaches (for instance, pairwise comparison between criteria). The calibration problem in our works is solved by fuzzy set elicitations and calibrations on exemplary data. The topic of utility elicitation has been successfully addressed in recent decades, and resulting in a vast number of techniques proposed. For further reading, we refer to [BB06, CP04, HH99, BBGP97, CGNS98, HF03, LJS03].

5.2 Hierarchical Modeling

The idea of hierarchical modeling is of course not new, hierarchical modeling of a rating function is an intuitively appealing and commonly used strategy, which generally simplifies the overall evaluation of an alternative by rating different sub-criteria at first, and then aggregates these ratings afterward. This idea has been employed in many works, for purpose of utility modeling and decision making, for example, the AHP approach mentioned in the previous section has gained the most similarity with our approach, because both follow the sample proposal to construct the utility function, namely decomposing the utility function into sub-utility functions recursively. In the following, let us examine some different approaches in short:

- Regarding the idea of hierarchical modeling, the perhaps most well-known related approach is a framework for evaluating documents proposed in [Yag00], in which a document retrieval language is developed to enable user represent their requirements by using appropriate aggregation operators. This framework has been shown to support aggregation in a hierarchical structure based on the OWA operators, which allowed a linguistic specification on the interrelationship between the involved attributes. The hierarchical structure and use of OWA operator make this method very similar to our approach in this thesis. However, an FOT model generalizes this framework by allowing all fuzzy logic-based operators including the OWA operator, and employing fuzzy sets for basic evaluations instead of requiring them directly. From this point of view, our approach is more comprehensible and reasonable for evaluation. Another, but maybe more important, difference lies in the structure calibration on an FOT model, which adapts an FOT model according to exemplary data.

- Another related work in this regard is proposed in [BR88], in which a similar hierarchical structure is learned by analyzing the interrelationship between attributes or criteria. To aggregate criteria at interior nodes, an elementary decision table is assigned in a data-driven way. Unfortunately, such tables might be difficult to understand, so the whole approach can be criticized from an interpretable point of view. Beside that, the elementary decision table saved in interior nodes is usually very large, its space complexity grows quickly as the size of dataset increases.

- The tree structure has been often used for supporting the inference of an expert system [Nil71], which is very useful in representing a backward-chaining inference that tries to solve a problem by breaking it up into smaller problems and solving them individually. As an example, the AND-OR graph proposed in [LCK08] is used for making inferences in the expert system area, which can be seen as a specialized symbolic-reasoning technique to solve difficult decision making problem.
- The hierarchical fuzzy rule-based systems (HFRBS) haven been developed in [WCM06, CHZ01], by decomposing the fuzzy models into a number of simpler sub problems with smooth transitions between them, in which the simple building process of the fuzzy rule base is extended in a hierarchical way in order to make the system more accurate.
- In order to build an object recognition system, a learning algorithm is developed to give a membership value for recognized object in [WCH$^+$95], in which low-level membership values are combined through an and-or tree structure to give an overall membership value. Although the AND and OR operators used in this approach are fixed, an adaptation process (error back-propagation) is applied to adjust low-level membership functions based on training data, which is, to our interest, comparable with the calibration part in this thesis.

For further reading we refer to [Yag93, Car82, Mor94, Wel85].

5.3 Fuzzy Logic-based Operators in Decision Making

Regarding this point, a huge amount of work has been done in the field of fuzzy preference modeling and multi-criteria decision making, especially in relation with fuzzy sets and fuzzy aggregation operators. A review of all related work is clearly beyond the scope of this thesis, for a comprehensive overview see [GOY02, PR02, Rou97, PP03]. To the best of our knowledge, however, an approach directly comparable to our FOTs, including a means for model calibration, has not been proposed so far.

A purposeful approach for fuzzy decision making consists in the usage of aggregation procedures that realize the idea for compensation and compromise between conflicting criteria, which are usually made with the help of aggregation operators. The first ones introduced by L. Zadeh are for the logical operations AND, OR and NOT as extensions of the binary logical operations in his seminal paper [Zad65]. Later, some research works revealed that the degree of compensation through human aggregated criteria is not expressed only by these operators. Thus, many operators have been proposed to represent human decision making more accurately. Let us review several works in the following text:

- **Ordered weighted averaging** (OWA) : In [Yag88], a family of aggregation operators called OWA operator is proposed to realize trade-offs between criteria by allowing a compensation between ratings. These operators provide a natural framework to the inclusion of several types of behavioral properties [GOY02]. Another similar operator is suggested in [ZZ80].
- **Fuzzy integrals** : Extending the concept of Lebesgue integral, Sugeno has proposed the concept of fuzzy measure and fuzzy integral in [Sug74], called Sugeno integral. Later, another definition was proposed in [MS89] by using a concept introduced by Choquet in capacity theory, called Choquet integral.

- **Using type** 2 **Fuzzy Sets** : In [SF07], the author uses the synthesis of the tools of Type 2 and Level 2 fuzzy sets (see [Men95, MJ02]) to elaborate an appropriate method for aggregation of fuzzy values in multiple criteria decision making, which is successfully applied to solve a tool steel material selection problem.

- **Hierarchical aggregation operator** : In [Dyc85], a hierarchical aggregative operator (a more general "and-or" connective) is introduced by replacing the associativity axiom with autodistributivity, so that aggregation of information can be also achieved with the help of quasilinear means [Acz66].

Once the type of aggregation operators has been chosen for a particular application, the remaining task is to identify the parameters of the chosen operator if necessary. One of the backgrounds of this thesis is the issue of parameter estimation for aggregation operators, which has been addressed, though, for simpler types of decision models, especially for models using a single aggregation operator in many research works. For example, the problem of fitting parameters on the basis of exemplary outputs has been studied for weighted mean and OWA operators [FY98, Tor99], the WOWA(weighted OWA) operator [Tor04], the Choquet integral [MR00, WLW$^+$00], and the Sugeno integral. In [CHHV98] a new method to calculate weights for the OWA operators is proposed by using linguistic quantifiers that represent the concept of fuzzy majority. Besides, attempts have been made to identify the parameters of such models using other types of information, such as the so-called "orness" or degree of disjunction [FM01, FM03b] as well as preferences and order relations [CW85, MR05]. Another aggregation operator based on the Yager's family of t-norms is given in [Hau99], in which a procedure is described for controlling fuzziness in fuzzy arithmetic operators by fixing the parameter in the Yager's family of t-norms.

There is also a vast number of applications that makes use of aggregation operators, for instance, Kevin Woods et al. [WCH$^+$95] built an object recognition system by learning membership functions, J.-R. Chang suggested dynamic fuzzy OWA model for group multiple criteria decision making in [CHCC06], and another system ANFIS (Adaptive-Network-Based Fuzzy Inference System) is put forward by J.R. Jang in [Jan93]. Recently several methods have been published to extract a fuzzy system from artificial neural networks (ANN), which is made of fuzzy "IF…Then…" rules and equivalent to the functionality of ANN [KM05], these methods are generalized by using a new fuzzy operators in [Man07].

5.4 Calibration of a Network-like Structure

With respect to calibration of a network-like structure, the calibration of FOTs may of course remind of related hierarchical models such as artificial neural networks (ANN) because of the similarity in structure. There are, however, important differences between our approach and commonly used ANN models, such as multilayer perceptrons. In particular, ANNs typically have a single type of node (associated with an activation function), and only the weights on the edges are adapted. Moreover, the topology is usually not a tree; instead, in feed-forward nets, two successive levels are fully connected. Finally, ANNs do not offer an obvious logical interpretation. The latter disadvantage is to some extend avoided by neuro-fuzzy systems [NKaRK97] and related approaches such as ANFIS mentioned in previous section, in which activation functions are replaced by logical operators. The other differences, however, still remain also for these approaches.

Calibration is a common task for models in the network-like structure in order to make them more accurate, therefore many approaches have been proposed in numerous research works for this purpose. In the field of neural networks, these approaches can be divided into two main groups: The supervised (associative) learning, such as the well-known error back-propagation, gradient descent, error-correlation learning, or competitive learning etc; or the unsupervised (self-organization) learning, which tunes the network to perform some kind of data compression like dimensionality reduction or clustering. For a summary of these approaches we refer to [NKaRK97, Ser94, AB99].

In the field of neuro-fuzzy system, which differs from neural network in that fuzzy signals and/or fuzzy weights are employed, many calibration techniques are naturally extended from those based on neural network, such as the fuzzy back-propagation [NS01, HBC93], back-propagation on α-cuts [IOT92]. Furthermore a self-learning and tuning strategy based on reinforcements is discussed in [BK92, CC98] and a genetic algorithm has been applied to train a neural-fuzzy system in [ZH94, KW97]. A comprehensive overview on these approaches is given in [BH94].

6 Conclusions and Future Work

This chapter concludes this thesis. In Section 6.1, we briefly recapitulate our main results for modeling utility functions in the form of FOT models. In Section 6.2, we give an outlook on future research directions, which could extend our FOT models concerning several different aspects.

6.1 Conclusions

In this thesis, we have introduced fuzzy operator trees as a convenient tool for modeling utility functions. The key idea of this approach, which appears to be appealing from a modeling point of view, is to express a utility function in terms of a hierarchical structure by decomposing criteria into sub-criteria in a recursive way. The evaluation of sub-criteria can be combined by means of aggregation operators of different character: Conjunctive, averaging and disjunctive. Each sub-criterion is either a basic evaluation, which can be extracted directly with the help of a user, or aggregation of other sub-criteria in turn. Additionally, the linguistic hedge is applied in FOTs to weigh sub-criteria. A basic evaluation is associated with a fuzzy set, which gives a numeric degree in $[0, 1]$ to describe the relative satisfaction upon the basic evaluation. To this end, the utility of any object can be expressed as an aggregation over a set of basic evaluations, the structure of FOT determines in which way those basic evaluations are aggregated, as well as how exactly a criterion is calculated based on sub-criteria. On the one hand, an FOT has a tree-like structure, which allows a user to specify rating functions in a very intuitive and systematic way. On the other hand, as the components of FOTs (node and edge) are associated with fuzzy sets, fuzzy operators or linguistic hedges taken from fuzzy set theory, an FOT has an excellent interpretability and can easily be expressed in natural language, so that the human being can understand an FOT well.

Though the original motivation of developing FOTs comes from quality control, where FOTs are intended to replace the traditional quality assessment method, we like to emphasize that FOTs are much more general and can be applied for rating all sorts of things. Decomposing a criterion into several sub-criteria is a typical approach commonly used in practice in order to evaluate objects and an FOT can be easily constructed following this principle. Furthermore, many applications have a modular structure, like a technical product, whose components can be decomposed into subcomponents in a recursive way. From the modeling point of view, a component can be modeled in the form of an FOT involving subcomponents and, at the same time, be used as a (plug-in) subtree in a superordinate FOT at a higher lever.

The “divide-and-conquer” strategy underlying FOTs makes the assessment of very complex systems controllable.

To support a human expert in designing FOTs, several elicitation techniques are proposed to build FOTs in an intuitive and convenient way. Afterward, we bring up the idea of using FOTs in a machine learning context, namely as a model class underlying regression and (ordinal) classification problems. We address the problem of FOTs built by human expert, in which FOTs sometimes lack accuracy because disagreements arise between an empirical evaluation by a human expert and an estimated one by FOTs. In order to conquer this problem, we have developed a calibration method based on evolutionary strategies that fits the parameters of a (qualitative) model structure to a given set of training data, which ought to mimic the behavior of human expert as accurate as possible. Note that a training data can be constructed by collecting the exemplary evaluations made by human expert. The experimental results that we obtained are rather promising and show that the calibration method reliably optimizes even quite complex FOTs and is very powerful to solve this kind of problems. In terms of accuracy and running time, the calibration method shows a satisfying performance compared with several related techniques. Even on noisy and discrete data, which can often be observed in practice, the calibration method can tune the parameters involved in an FOT to fit a given training data successfully within an acceptable time.

Subsequently, we have used FOTs for modeling an evaluation strategy in the game of poker, in order to investigate the performance of our method in this case, a fair comparison has been made between FOT and several well-known approaches, which work in a purely data-driven way. The experimental results show that our method is powerful enough to be applied in practice, in which an FOT built by a human expert has a very strong inductive bias, so that the risk of overfitting for a small amount of training data can be reduced significantly. Although, as the amount of provided training data increases, the models delivered by other approaches can achieve a slightly higher accuracy and might have less risk of underfitting, these models are becoming quite complex and difficult to understand, in contrast to our FOT model, which has a fixed structure and therefore a relatively small, constant complexity. Amongst other, an FOT model provides an excellent interpretability independent of the amount of provided training data, which becomes extremely important in several applications such as quality control or classification in medicine, in these fields, no human expert will trust in a model with very complex and instable structure.

Finally we have addressed a cost issue by applying FOTs, since in practice a precise evaluation is not always necessary and the basic evaluations of FOTs may have different evaluation costs, which demand an evaluation plan to save evaluation costs efficiently. We have proposed two algorithms to predict optimal evaluation plans. As the experimental results show, the evaluation plans determined by both algorithms require significant less evaluation costs than other evaluation plans generated in a random or greedy manner.

6.2 Future Work

In this section, we sketch several interesting issues that remain to be addressed in future work. Our intention is not to present complete solutions, but rather to convey a feeling for the wide scope of extensions and applications based on the techniques introduced in this thesis, a more detailed elaboration of the presented approaches remains to be done in future work.

Structure Adaptation From a machine learning point of view, it may also be interesting to adapt the structure of an FOT, or at least parts thereof, in addition to the model parameters. In this thesis, we restrict ourselves to use a fixed structure of FOTs, which is built with the help of human experts. During the calibration process, the structure of FOTs (qualitative part) is viewed as constant and can not be tuned as the model parameters. In practice, the elicitation process to build an FOT with the help of a human expert is sometimes not trivial, in cases where uncertainty might exist. One possible approach to this problem is to parameterize the structure with constraints and employ a discrete optimizer to fit given training data, such as genetic algorithms [TF02].

Preparation of Training Data Furthermore an FOT model can be treated as a tool to interpret the decision process with the help of fuzzy theory, which tries to extract the preference of experts over a set of alternatives. In order to calibrate an FOT, a set of alternatives reflecting the preference of experts is supplied as training data. In this thesis, we require a training data labeled with single assessment, which can be a numeric value, or a class label etc. According to [CHHV98], there are three different ways to supply the information about the alternatives:

- As *an utility function*: In this case an expert supplies a real evaluation for each alternative with a real number indicating the quality of that alternative according to his point of view.
- As *a fuzzy preference relation*: In this case an expert supplies a fuzzy binary relation over the set of alternatives, reflecting the degree to which an alternative is preferred to another.
- As *a preference ordering of the alternative*: In this case the alternatives are ordered from the best to the worst, namely labeled with a ranking, without any other supplementary information.

Apparently our method to calibrate FOTs belongs to the first category, whereas the last two categories are not touched in this thesis. How to utilize different kinds of information about the alternatives is another challenge for extending an FOT model.

Ordinal Classification Problem In the machine learning context, FOTs can be used to solve ordinal classification problems. In fact there are many applications of learning methods, for example in the field of recommender systems [IS85], in which the model to be induced can generally be considered as a kind of (discrete) utility function. An extension of this work can be addressed in using FOTs as a kind of learning tool, where the ability of FOTs to incorporate background knowledge becomes especially advantageous.

A Appendix: List of t-norms and t-conorms

Table A.1: List of t-conorm

Name	Function
Algebraic	$S(x,y) = x + y - x \cdot y$
Dombi	$S_\lambda(x,y) = \frac{1}{1+((\frac{1}{x}-1)^{-\lambda}+(\frac{1}{y}-1)^{-\lambda})^{-\frac{1}{\lambda}}}$ $\lambda \in (0,\infty)$
Drastic (Weak)	$S(x,y) = \begin{cases} y & \text{if } x = 0, \\ x & \text{if } y = 0, \\ 1 & \text{otherwise} \end{cases}$
Drastic	$S(x,y) = \begin{cases} 0 & \text{if } x = 0 \text{ and } y = 0, \\ 1 & \text{otherwise} \end{cases}$ the biggest t-conorm
Dubois & Prade	$S_\alpha(x,y) = \frac{x+y-x\cdot y-min(x,y,1-\alpha)}{max(1-x,1-y,\alpha)}$ $\alpha \in [0,1]$
Einstein	$S(x,y) = \frac{x+y}{1+y\cdot x}$
Frank	$S_s(x,y) = 1 - log_s(1 + \frac{(s^{1-x}-1)\cdot(s^{1-y}-1)}{s-1})$ $s \in (0,+\infty)$, $s \neq 1$
Hamacher	$S_r(x,y) = \frac{x+y-(2-r)\cdot x\cdot y}{1-(1-r)\cdot x\cdot y}$ $r \geq 0$
Lukasiewicz	$S(x,y) = min(1, x+y)$ also the called Bold t-conorm
Maximum	$S(x,y) = max(x,y)$ also called the Zadeh t-conorm, the least t-conorm
Schweizer & Sklar	$S_p(x,y) = 1 - max(0,(1-x)^{-p} + (1-y)^{-p} - 1)^{-\frac{1}{p}}$ $p \in (-\infty,+\infty)$, $p \neq 0$
Weber	$S_p(x,y) = min(1, x+y+(p-1)\cdot x\cdot y)$ $p \geq 0$
Yager	$S_w(x,y) = min(1,(x^w+y^w)^{\frac{1}{w}})$ $w \in (0,+\infty)$

Table A.2: List of t-norm

Name	Function
Algebraic	$T(x,y) = x \cdot y$ also called product t-norm
Dombi	$T_\lambda(x,y) = \frac{1}{1+((\frac{1}{x}-1)^\lambda + (\frac{1}{y}-1)^\lambda)^{\frac{1}{\lambda}}}$ $\lambda \in (0,+\infty)$
Drastic (Weak)	$T(x,y) = \begin{cases} y & \text{if } x = 1, \\ x & \text{if } y = 1, \\ 0 & \text{otherwise} \end{cases}$
Drastic	$T(x,y) = \begin{cases} 1 & \text{if } x = 1 \text{ and } y = 1, \\ 0 & \text{otherwise} \end{cases}$ the smallest t-norm
Dubois & Prade	$T_\alpha(x,y) = \frac{x \cdot y}{max(x,y,\alpha)}$ $\alpha \in [0,1]$, $\alpha = \begin{cases} 0 & \text{Minimum t-norm} \\ 1 & \text{Algebraic t-norm} \end{cases}$
Einstein	$T(x,y) = \frac{x \cdot y}{2-x-y+x \cdot y}$
Frank	$T_s(x,y) = log_s(1 + \frac{(s^x-1)\cdot(s^y-1)}{s-1})$ $s \in (0,+\infty)$, $s \neq 1$
Hamacher	$T_r(x,y) = \frac{x \cdot y}{r+(1-r)\cdot(x+y-x \cdot y)}$ $r \geq 0$, $r = \begin{cases} 1 & \text{Algebraic t-norm} \\ 2 & \text{Einstein t-norm} \\ \infty & \text{Drastic t-norm} \end{cases}$
Lukasiewicz	$T(x,y) = max(0, x+y-1)$ also called the bold t-norm
Minimum	$T(x,y) = min(x,y)$ also called the Zadeh t-norm, the greatest t-norm
Schweizer & Sklar	$T_p(x,y) = max(0, x^{-p} + y^{-p} - 1)^{-\frac{1}{p}}$ $p \in (-\infty,+\infty)$, $p \neq 0$
Weber	$T_p(x,y) = max(0, x \cdot y - p \cdot (1-x) \cdot (1-y))$ $p \geq 0$, $p = \begin{cases} 0 & \text{Algebraic t-norm} \\ 1 & \text{Lukasiewicz t-norm} \\ \infty & \text{Drastic t-norm} \end{cases}$
Yager	$T_w(x,y) = 1 - min(1, ((1-x)^w + (1-y)^w)^{\frac{1}{w}})$ $w \in (0,+\infty)$

B Appendix: A Java Implementation

In this thesis, we have developed the system to provide an interactive framework for building, visualizing, optimizing and analyzing FOTs, which is compressed in a Java package *robq.jar* and named as *RobQ* for "Robot-based Quality Assessment Tool"[1]. This system is based on a Java data mining package *Weka* [FHT97], which contains a complete framework to arrange experiments, compare data mining algorithms, manage data sets, and so on. In following, we describe the class structure of the RobQ system in Section B.1, and introduce the functionalities of the RobQ system in Section B.2.

The RobQ system is implemented in following environments: Java(TM) 2 Runtime Environment 1.5.0; Eclipse SDK 3.2.2; Weka 3.4.3.

B.1 The Class Structure

The RobQ system consists of five major packages, as Figure B.1 illustrates, they are:

1. **Operator Package** This package is a collection of fuzzy components to build FOTs, as well as the parameter classes used in fuzzy components, loss functions and classes for evaluation plans. Figure B.2 shows the structure of the operator package. The involved classes can be categorized into following 3 groups:

   (a) **Norm** and **FuzzySet** : The core of operator package is the operator interface, which defines the common methods of fuzzy components of FOTs. Not only fuzzy sets, but also norms and linguistic hedges used in constructing FOTs can be generalized as mapping, which take the outputs of child operators as input, and return numeric output to the parent operator in turn. Furthermore the sub interfaces of fuzzy set and norm intend to generalize the common properties of fuzzy set and norms, notice that the linguistic hedge is built into operator with one additional parameter upon output of operator. Extending the norm interface, the interfaces of t-norm, t-conorm and mixnorm are defined.

   The aforementioned interfaces give a general framework to define any

[1] free available under *http://www.mathematik.uni-marburg.de/~yi*

Figure B.1: Class structure of RobQ package

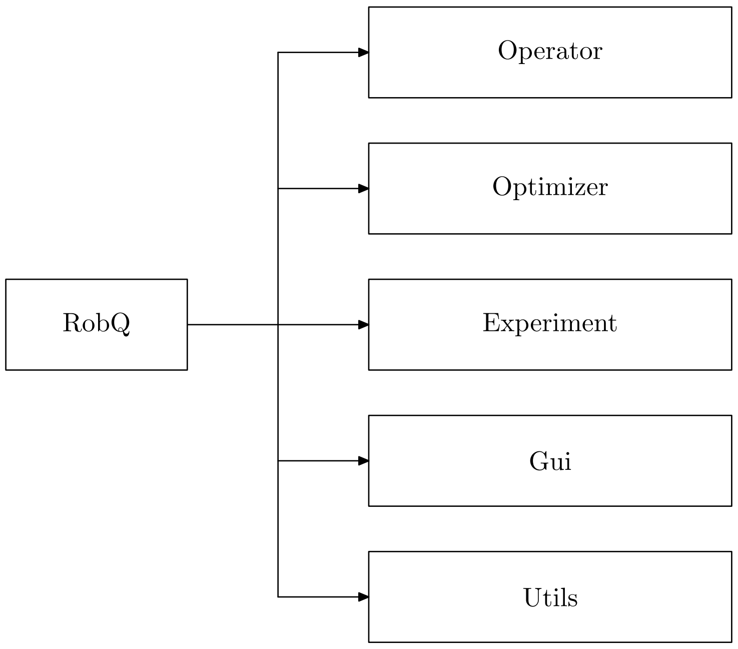
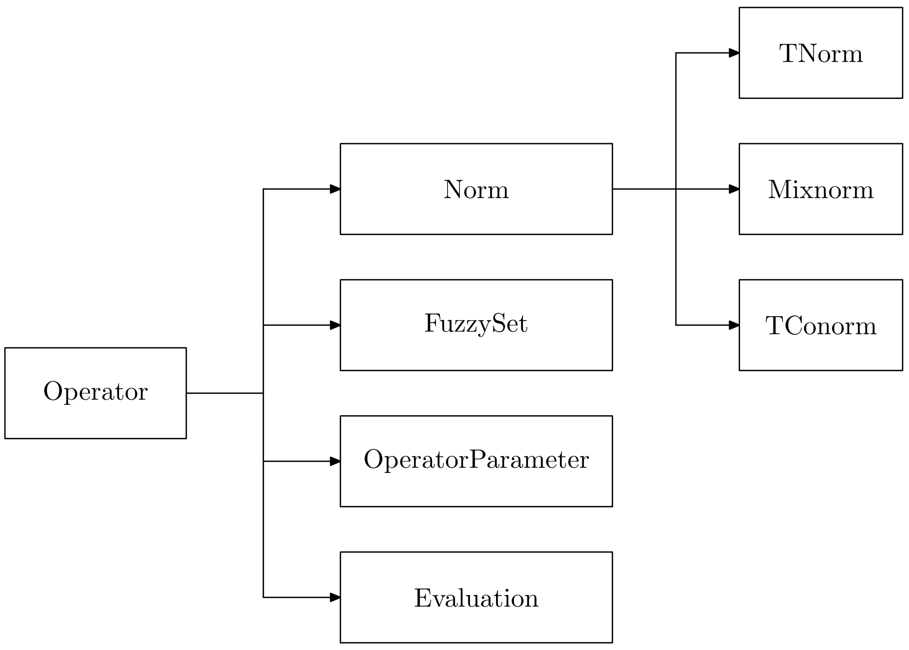


Figure B.2: Operator package of RobQ package



aggregation operator, so that one can easily develop new operators. All common methods and member fields are collected in a direct inherited abstract "Operator" class, so that just a little work is needed to define an operator, that is, to characterize its distinct fields or functions.

(b) **Operator Parameter** : Since the optimization on FOTs bases mainly on specifying parameters of operators in FOTs, the operator parameter interface summarizes the operations on operator parameters. A numeric parameter, no matter involved in fuzzy sets or norms, or linguistic hedges, is characterized with its domains and excluding points. For example, a valid parameter for the Hamacher t-norm can be defined as:

$$r := \text{OperatorParameter}(0, +\infty, \{0\})$$

namely it has a domain $[0, +\infty]$ excluding 0.

All common operations on parameters are generalized in this class, such as the delta rule of gradient descent methods, or mutation operation in evolution strategies.

(c) **Evaluation Plan** : For the purpose of evaluation cost minimization on FOTs, several related classes are developed here to present an evaluation plan, or a distribution of operator, etc.

Beside these, several utility classes for operators are developed here:

- The *Operator generator* class helps to generate a random operator under given limitations. For example, in order to generate artificial training data, this class is used to generate an operator randomly with predefined complexity.
- The *Data generator* class, as its name indicates, is able to generate a random training data with given FOTs and other limitations, such as built-in error.
- The *Loss Function* class generalizes the loss functions for the purpose of comparison on FOTs with discrete outputs.
- The *Differentiable* interface collects the common methods of differentiable operators, which is specially designed for the gradient descent methods.
- The *FuzzyOperatorTree* class enables to save an FOT in the form of XML file, and load an FOT saved in an XML file reversely.

2. **Optimizer Package** The Optimizer package is a collection of implementations of calibration techniques on FOTs. They are categorized into a "genetic" package, which is devoted to the class of evolution strategies, and a "gradient-descent" package, which includes the gradient descent class and its variants, as well as the simulated annealing class.

   Under the "genetic" package, there are still further packages to generalize the crossover and mutation operation in the evolution strategies, as well as the individual interface. The Optimizer package provides a flexible platform, on which one can easily integrate an optimization technique into the RobQ system.

3. **Experiment Package** Like the experiment package in Weka, the Experiment package here provides a complete framework to arrange the calibration experiments on FOTs. As an extension of the Weka package, we have developed following classes:

(a) The *Experiment* class overwrites the default experiment class in Weka for special tasks on FOTs, such as the experiments to evaluate the calibration techniques or the algorithms for minimizing the evaluation cost.

(b) The *WekaAnalyse* class analyses the experimental results of FOTs. As the Weka package provides a framework to run batch of experiments, record the output of experiment into text files, and analyze experiment output visually, this class takes over the analyse process of Weka on the experiment output, and integrates the special tasks regarding respective goals, such as outputting the result into text format, preparing further graphical representation on experiment output, and so on.

4. **GUI Package** The GUI package in RobQ collects the classes and interfaces for the graphical interactive tool on FOTs, which is described in more detail later in Section B.2.1.

5. **Utils Package** This package contains the utility classes of Weka involved in RobQ, so that the RobQ system becomes independent of the Weka system.

B.2 The Main Functionalities

The RobQ system enables the user to create, run, modify, and analyze FOTs in a convenient manner under both graphical and command modules. In this section, we describe how to run the RobQ system under both modules in detail.

B.2.1 Graphical Module

The most convenient way to explore the functionalities of the RobQ system is to enter its graphical module, which is based on the open source widget toolkit SWT. If the computer has a Java environment (both under MS windows and Linux), one can enter the graphical module by double-clicking the RobQ package usually; otherwise the following command line is needed:

```
java -jar robq.jar
```

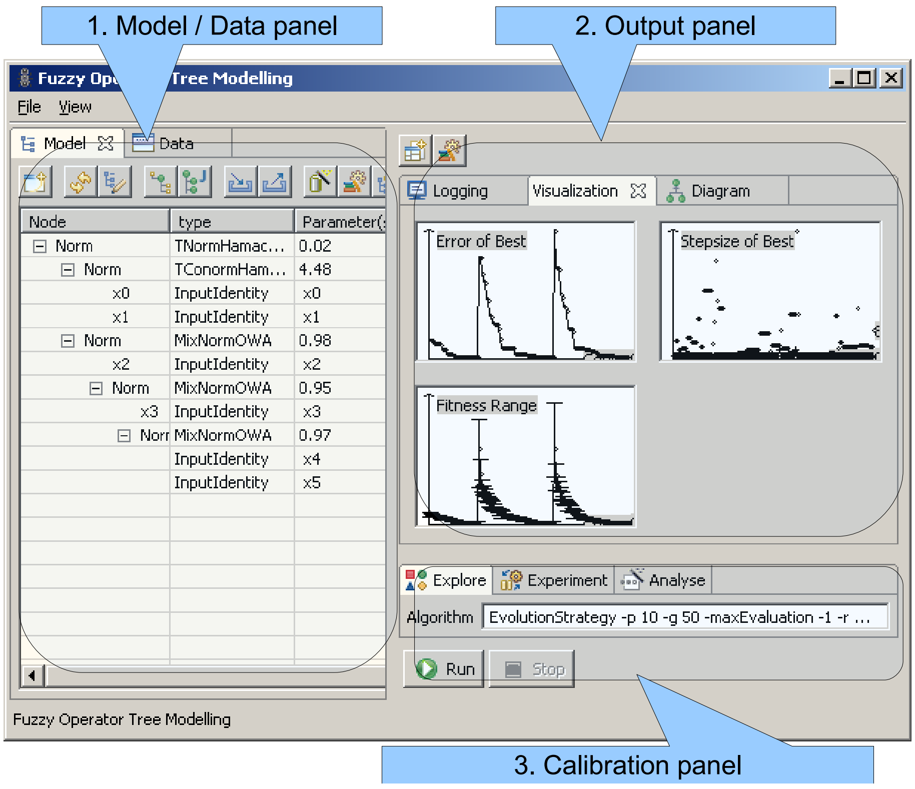
The main panel of the RobQ system is shown in Figure B.3. We introduce its three major panels in the following text.

B.2.1.1 The Model/Data Panel

The model/data panel consists of a model panel and a data panel. The former puts the functionalities for building, modifying FOTs together, and the data panel manages training data.

As one can see, an FOT is visualized in the model panel using the TableTree, where each operator object takes a row in the nested table and consists of three columns: “Node”, “Type” and “Parameter”. The nice property of the TableTree composite lies in that the first column demonstrates the hierarchical relationship between operators, while the rest columns describe the detailed information on related operators. The toolbar at the top of the model panel contains a set of functional buttons, they are:

- creates a new FOT by removing the current operator and inserting one operator. To guarantee the completeness of an FOT, only “Create” and “Modify” operations are allowed, while “Add”(or “Insert") and “Remove” operations are



Node	type	Parameter(
Norm	TNormHamac...	0.02
Norm	TConormHam...	4.48
x0	InputIdentity	x0
x1	InputIdentity	x1
Norm	MixNormOWA	0.98
x2	InputIdentity	x2
Norm	MixNormOWA	0.95
x3	InputIdentity	x3
Norr	MixNormOWA	0.97
	InputIdentity	x4
	InputIdentity	x5

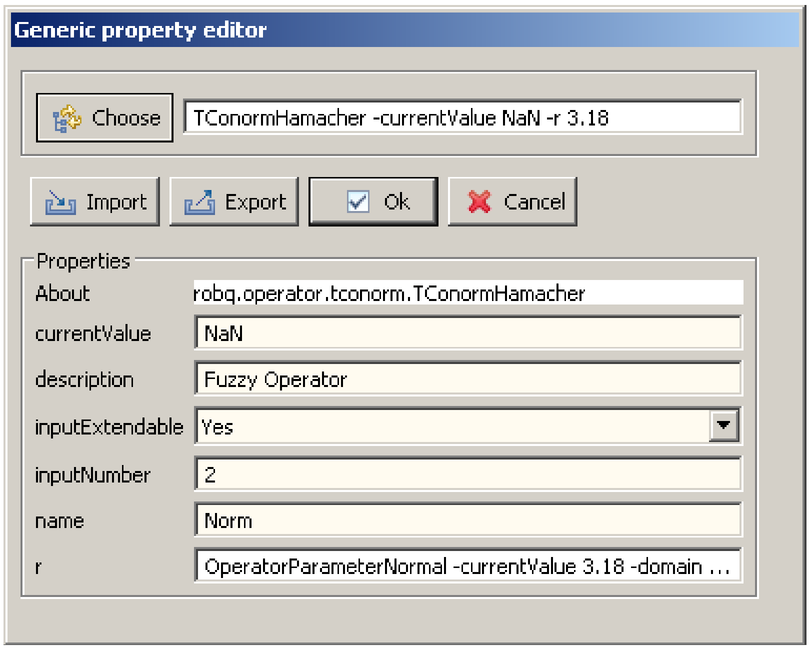
Algorithm EvolutionStrategy -p 10 -g 50 -maxEvaluation -1 -r ...

Figure B.3: Graphical interface of RobQ package

not provided, since an FOT is only complete, if all its interior nodes are norms, and its leaf nodes are fuzzy sets. The “Create” and “Modify” operations generate or change an operator and ensure that the current operator is complete, that is, if the current operator is a norm, then the (default number of) leaf nodes are automatically inserted. Otherwise its children nodes, if exist, are removed.

- modifies the property of selected operator. Generally to design the modification panel is very time-consuming, since objects usually have different properties, sometimes its property is in turn an object. The idea of “generic object editor” has been emerged originally in Weka to overcome this problem. The generic object editor in Weka gives a framework to define the properties of objects, so that a uniform property editor can be applied in any objects following this framework. Following this idea, we have developed an extended *generic property editor*, in which not only numeric and text properties, but also nominal and object ones can be modified. Figure B.4 illustrates a generic property editor to modify the properties of a Hamacher t-conorm, where the parameter specifications on a Hamacher t-norm are listed and ready to be modified. As one can see, the text and numeric properties are composed in the form of text editor, while nominal property (like “inputExtendable" here) is composed in the form of list editor. The built-in object property (the parameter *r* here) is in turn modifiable by embedding another generic property editor recursively. The common head of a generic property editor contains a “Choose” button, which lists all alternative objects to current one, so that one can change the type of operator (for instance changing to a t-norm or fuzzy set in this example is possible), and a description label summarizes current operator. With further four buttons one can import or export the properties in the form of text file, confirm or cancel the modifications.

To take the benefit of the generic property editor, one just needs to declare a property with following methods:



TConormHamacher -currentValue NaN -r 3.18

Properties

About	robq.operator.tconorm.TConormHamacher
currentValue	NaN
description	Fuzzy Operator
inputExtendable	Yes
inputNumber	2
name	Norm
r	OperatorParameterNormal -currentValue 3.18 -domain ...

Figure B.4: Generic Property Editor on a Hamacher t-conorm

- the $getXXX()$ method,
- the $setXXX()$ method and
- the $xXXTipText()$ method.

where XXX stands for the property name, and xXX for the property name with first letter in lowercase. Then the modification on XXX is arranged automatically with the help of the generic property editor, no further work is necessary, as long as XXX is a numeric, text, nominal or object property.

- and randomize the involved parameters in current operator or FOT, which is specially desired for generating artificial training data or initializing FOTs before the calibration process.

- and allow user to import an FOT saved in an XML file or export an FOT into an XML file reversely.

- generates an artificial training data based on current FOT, while sets the data generator process.

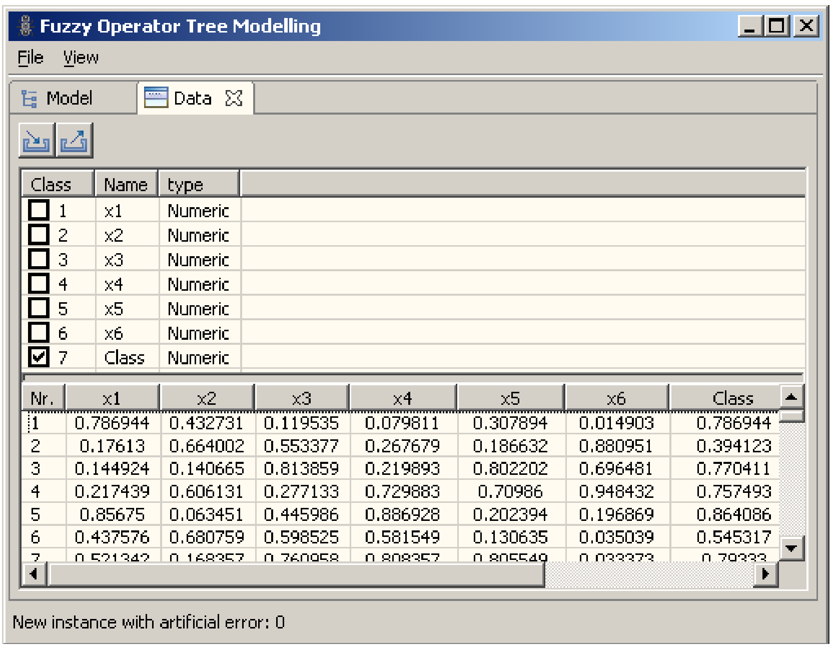
The Data panel provides a quick view of data files in the form of $.arff$, which is the standard format in the Weka package. In the Data panel, user can browse, import and export training data, as well as modify the data, set the class attributes, etc. Figure B.5 demonstrates a data panel.

B.2.1.2 The Output Panel

The output panel consists of following parts:

1. The Logging panel records the runtime information of operations called in the following calibration panel.

2. The Visualization panel shows the intermediate results during the calibration process graphically by drawing the important measurements in 2D spaces.



Class	Name	type
☐ 1	x1	Numeric
☐ 2	x2	Numeric
☐ 3	x3	Numeric
☐ 4	x4	Numeric
☐ 5	x5	Numeric
☐ 6	x6	Numeric
☑ 7	Class	Numeric

Nr.	x1	x2	x3	x4	x5	x6	Class
1	0.786944	0.432731	0.119535	0.079811	0.307894	0.014903	0.786944
2	0.17613	0.664002	0.553377	0.267679	0.186632	0.880951	0.394123
3	0.144924	0.140665	0.813859	0.219893	0.802202	0.696481	0.770411
4	0.217439	0.606131	0.277133	0.729883	0.70986	0.948432	0.757493
5	0.85675	0.063451	0.445986	0.886928	0.202394	0.196869	0.864086
6	0.437576	0.680759	0.598525	0.581549	0.130635	0.035039	0.545317
7	0.521342	0.168357	0.760958	0.808357	0.805549	0.033373	0.79333

New instance with artificial error: 0

Figure B.5: The Data panel of RobQ package

3. The Diagram panel shows the current FOT in a real tree-like structure, in which the color depth corresponds the parameter specifications, as Figure B.6. For example, the maximal t-norm (the minimum t-norm) is colored with green, which is at the same time the minimal mixnorm. On the other hand, a minimal t-norm is colored with white. The diagram gives a general representation of an FOT in a "fuzzy" way.

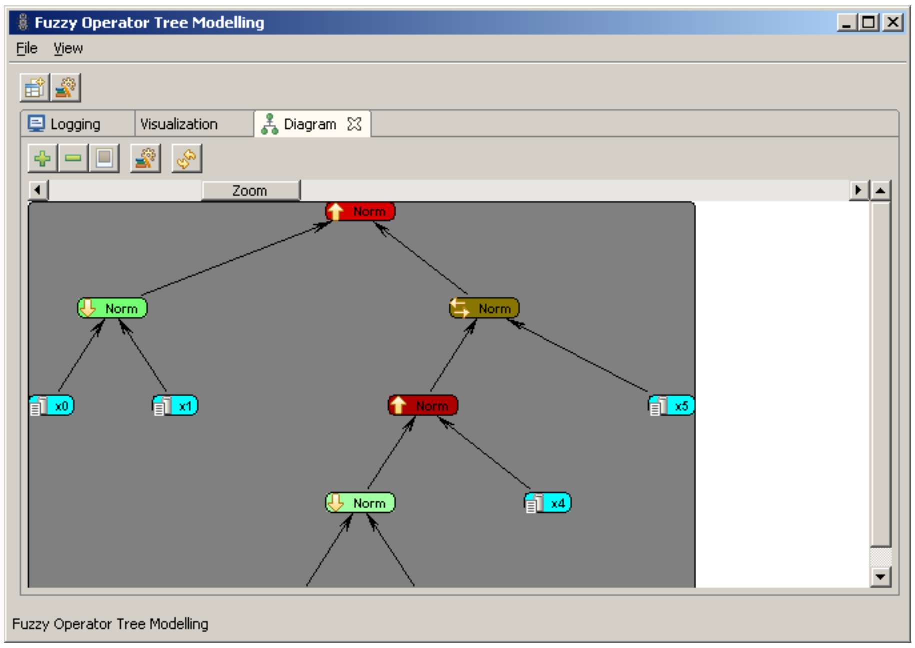


Figure B.6: Diagram of an FOT

B.2.1.3 The Calibration Panel

This panel is especially developed for the experiments of calibration of FOTs by specifying the experiment or analyser class with the help of generic property editor. It consists of following three subpanels:

1. Under the Explore panel , one can arrange a single experiment by specify-

ing an involved calibration technique and setting its parameters. The intermediate results of the calibration process is shown in the Logging and Visualization panels.

2. Under the Experiment panel , one can arrange a set of experiments by specifying the modified experiment class, for a more flexible way to arrange experiments see Section B.2.2.

3. The Analyse panel is designed to analyze the experimental results, the analyse output is written into the Logging panel. Inherited from the Weka package, one can also specify the format of analyse output, the columns to be compared, and so on.

B.2.2 Command Module

While under the graphical module user can make use of the RobQ system in an interactive way, the command module is desired for arranging experiments and analyzing the experiment outputs in a professional, efficient way. Without loading the graphical components, running the RobQ system under the command module requires significant less memory and disk space. Moreover, all mentioned tasked in this thesis are available under the command module, under the graphical module only several tasks like calibration is allowed. Actually all experiments involved in this thesis are carried out under the command module.

Under the command module, one can explore the RobQ system by specifying proper parameters. A typical command under the command module looks like:

```
java -cp robq.jar -mx1024m robq.experiment.Experiment
-L 1 -U 1
-D robq.experiment.resultListener.Instances
-O optimizer.txt -T 50 -C 10 -I 200
-- -O 0620_c010.arff
-W robq.experiment.splitEvaluator.Optimizer
```

where the first line gives the classpath (*robq.jar* here), the required visual memory (1024*MB*) and the main class (*Experiment*), the last parameter can be replace with other class names for different tasks, for instance use

```
robq.experiment.analyser.WekaAnalyse
```

to analyse the experiment outputs. From the second line the parameter specification is given pairwisely, in this example they are:

- L and U the lower and upper run number of experiments, which determines how many times each experiment would be repeated. After applying the "restart" technique, these both parameters are set to be identical. That is, the experiment under each configuration would be carried out once.

- D indicates the format of experiment output, in this case it is in turn a classname standing for the standard experiment output ($.arff$ file).

- O gives the configuration filename for optimizer to be investigated.

- C is the complexity of randomly generated FOTs.

- *I* is the number of instances in an artificial training data.
- $--$ is the separator to divide the parameters between different classes, like in this case, the second parameter *O* is set to the experiment output as the output filename.
- *W* indicates the evaluation class to generate the experiment output.

To show all available parameters, one can simply invoke a class without special parameter. For example, use

```
java -cp robq.jar -mx1024m robq.experiment.Experiment
```

to get a list of all available parameters for the Experiment class.

List of Figures

List of Tables

List of Algorithms

Index

Bibliography

[AB99] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, New York, 1999.

[Acz66] J. Aczel. *Lectures on Functional Equations and their Applications*. Academic Press, 1966.

[Ana93] P. Anand. *Foundations of Rational Choice under Risk*. Oxford University Press, Oxford, 1993.

[Bäc96] T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Oxford, UK, 1996.

[BB03] T. Bartz-Beielstein. Experimental analysis of evolution strategies - overview and comprehensive introduction. Technical Report Reihe CI 157/03, Universität Dortmund, 2003.

[BB05] T. Bartz-Beielstein. *New experimentalism applied to evolutionary computation*. PhD thesis, University Dortmund, 2005.

[BB06] D. Braziunas and C. Boutilier. Preference elicitation and generalized additive utility. In *Association for the Advancement of Artificial Intelligence (AAAI)*. AAAI Press, 2006.

[BBGP97] C. Boutilier, R. Brafman, C. Geib, and D. Poole. A constraint-based approach to preference elicitation and decision making. In J. Doyle and R. H. Thomason, editors, *AAAI Spring Symposium on Qualitative Preferences in Deliberation and Practical Reasoning*, pages 19–28, Menlo Park, California, 1997.

[BBLP05] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuß. Sequential parameter optimization. In B. McKay and colleagues, editors, *Proc. Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, volume 1, pages 773–780, Piscataway NJ, 2005. IEEE Press.

[BBPV04] T. Bartz-Beielstein, K. E. Parsopoulos, and M. N. Vrahatis. Analysis of particle swarm optimization using computational statistics. *Proc. Int. Conf. Numerical Analysis and Applied Mathematics (ICNAAM)*, pages 34–37, 2004.

[Bey01] H. Beyer. *The theory of evolution strategies*. Springer, Heidelberg, 2001.

[BH94] J. J. Buckley and Y. Hayashi. Fuzzy neural networks: A survey. *Fuzzy Sets and Systems*, 66(1):1–13, 1994.

[BH07] K. Brinker and E. Hüllermeier. Case-based label ranking. In *Proc. of the 17th European Conference on Machine Learning (ECML'06), 20th Int. Joint Conf. on Artificial Intelligence*, Hyderabad, India, 2007.

[BHS98] S. Barbera, P. J. Hammond, and C. Seidl, editors. *Handbook of Utility Theory. Volume 1 Principles*. Kluwer Academic Publishers, Dordrecht, 1998.

[BHS04] S. Barbera, P. J. Hammond, and C. Seidl, editors. *Handbook of Utility Theory. Volume II Extensions*. Kluwer Academic Publishers, Dordrecht, 2004.

[BK92] H. Berenji and P. Khedkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Trans. on Neural Networks*, 3(5):724–740, 1992.

[BKKN03] C. Borgelt, F. Klawonn, R. Kruse, and D. Nauck. *Neuro-Fuzzy-Systeme, Von den Grundlagen Neuronaler Netze zu modernen Fuzzy-Systemen*, volume 3. Vieweg-Verlag, Wiesbaden, 2003.

[Bly02] J. Blythe. Visual exploration and incremental utility elicitation. In *Association for the Advancement of Artificial Intelligence*, pages 526–532, 2002.

[BNKF01] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming - An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, dpunkt.verlag, 3rd edition, 2001.

[BR88] M. Bohanec and V. Rajkovic. Knowledge acquisition and explanation for multi-attribute decision making. *In 8th Int. Workshop on Expert Systems and their Applications*, pages 59–78, 1988.

[BR90] M. Bohanec and V. Rajkovic. Dex: an expert system shell for decision support. *Sistemica*, 1(1):145–157, 1990.

[BS95] B. Bassana and M. Scarsini. On the value of information in multi-agent decision theory. *Journal of Mathematical Economics*, 24(6):557–576, 1995.

[BS02] H. Beyer and H. Schwefel. Evolution strategies - a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[BSW89] A. G. Barto, R. S. Sutton, and C. J. Watkins. Learning and sequential decision making. Technical report, University of Massachusetts, Amherst, MA, USA, 1989.

[BT02] T. Bilgic and I. B. Türksen. *Measurement of membership functions: theoretical and empirical work*, volume Fundamentals of Fuzzy Sets of *The Handbooks of Fuzzy Sets Series*, chapter Fuzzy Sets, pages 195–230. Kluwer Academic Publishers, Boston/London/Dordrecht, 2002.

[Cam95] C. Camerer. *The Handbook of Experimental Economics*, chapter Individual Decision Making, pages 587–703. Princeton University Press, Princeton, 1995.

[Car82] C. Carlsson. Realism in hierarchical modeling: A fuzzy system approach. In *Int. Working Conf. on Model Realism*, pages 5–21, 1982.

[CC98] H. Chung and C. Chiang. A self-learning and tuning fuzzy logic controller based on genetic algorithms and reinforcements. *Int. Journal of Intelligent Systems*, 12(9):673–694, Dec 1998.

[CGNS98] U. Chajewska, L. Getoor, J. Norman, and Y. Shahar. Utility elicitation as a classification problem. In *Proc. of the 14th Conf. Uncertainty in Artificial Intelligence (UAI'98)*, pages 79–88, San Francisco, 1998. Morgan Kaufmann Publishers.

[Cha87] J. L. Chameau. Membership functions part I: Comparing method of measurement. *Int. Journal of Approximate Reasoning*, 1:287–301, 1987.

[CHCC06] J. Chang, T. Ho, C. Cheng, and A. Chen. Dynamic fuzzy owa model for group multiple criteria decision making. *Soft Computing*, 10(7):543–554, 2006.

[CHHV98] F. Chiclana, F. Herrera, and E. Herrera-Viedma. Integrating three representation models in fuzzy multipurpose decision making based on fuzzy preference relations. *Fuzzy Sets and Systems*, 97(1):33–48, Jul 1998.

[CHZ01] O. Cordon, F. Herrera, and I. Zwir. Fuzzy modeling by hierarchically built fuzzy rule bases. *Int. Journal of Approximate Reasoning*, 27(1):61–93, 2001.

[CK93] S. Cho and J. H. Kim. Rapid backpropagation learning algorithms. *Circuits, Systems, and Signal Processing*, 12(2):155–175, 1993.

[CL03] C. Chen and B. Liu. Linguistic hedges and fuzzy rule based systems. In J. Casillas, O. Cordon, F. Herrera, and L. Magdalena, editors, *Accuracy improvements in linguistic fuzzy modeling: studies on fuzziness and soft-computing*, volume 129. springer, heidelberg, 2003.

[Cle95] R. T. Clemen. *Making Hard Decisions: An Introduction to Decision Analysis*. Duxbury Press, Pacific Grove, California, 1995.

[CMG$^+$93] D. E. Coleman, D. C. Montgomery, B. H. Gunter, G. J. Hahn, P. D. Haaland, M. A. O'Connell, R. V. Leon, A. C. Shoemaker, and K. Tsui. A systematic approach to planning for a designed industrial experiment. *Technometrics*, 35(1):1–27, 1993.

[Coc02] M. D. Cock. Linguistic hedges: a quantifier based approach. In A. Abraham, J. Ruiz del Solar, and M. Köppen, editors, *Soft Computing Systems - Design, Management and Applications*, volume 87 of *Frontiers in Artificial Intelligence and Applications*, pages 142–152. IOS Press, 2002.

[Coe06] C. Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE computational intelligence*, 1(1):28– 36, Feb 2006.

[CP04] L. Chen and P. Pu. Survey of preference elicitation methods. Technical report, Swiss Federal Institute of Technology in Lausanne, Jul 2004.

[CR06] A. Chandramohan and M. V. C. Rao. Novel, useful, and effective definitions for fuzzy linguistic hedges. *Discrete Dynamics in Nature and Society*, pages 1–13, 2006.

[CW85] E. U. Choo and W. C. Wedley. Optimal criterion weights in repetitive multicriteria decision-making. *The Journal of the Operational Research Society*, 36(11):983–992, 1985.

[DK08] G. N. Diubin and A. A. Korbut. Greedy algorithms for the minimization knapsack problem: Average behavior. *Int. Journal of Computer and Systems Sciences*, 47(1):14–24, Feb 2008.

[DP80] D. Dubois and H. Prade. New results about properties and semantics of fuzzy set-theoretic operators. In P. Wang and S. Chang, editors, *Fuzzy Sets: Theory and Applications to Policy Analysis and Information Systems*, pages 59–75. Plenum, New York, 1980.

[DP93] D. Dubois and H. Prade. Fuzzy sets and probability: misunderstandings, bridges and gaps. In *Proc. of the 2nd IEEE Conf. on Fuzzy Systems*, pages 1059–1068, 1993.

[DS05] A. Detwarasiti and R. D. Shachter. Influence diagrams for team decision analysis. *Decision Analysis*, 2(4):207–228, 2005.

[Dyc85] H. Dyckhoff. Basic concepts for a theory of evaluation hierarchical aggregation via autodistributive connectives in fuzzy set theory. *European Journal of Operational research*, 20:221–233, 1985.

[Dye90] J. S. Dyer. Remarks on the analytic hierarchy process. *Management Science*, 36(3):249–258, 1990.

[Fah88] S. Fahlman. An empirical sudy of learning speed in back-propagation networks. Technical report, Carnegie Mellon University, 1988.

[FHT97] E. Frank, M. Hall, and L. Trigg. Weka: The waikato enviroment for knowledge analysis, Jun 1997.

[Fis69] P. C. Fishburn. *Utility Theory for Decision Making*. Robert E. Krieger Publishing Co., Huntington, New York, 1969.

[FK01] J. Fürnkranz and M. Kubat, editors. *Machines that learn to play games*. Nova Science Publishers, Inc., Commack, NY, USA, 2001.

[FM01] R. Fullér and P. Majlender. An analytic approach for obtaining maximal entropy owa operator weights. *Fuzzy Sets and Systems*, 124(1):53–57, 2001.

[FM03a] B. Fiehn and J. P. Müller. User adaptive matchmaking. *Proc. of the Künstliche Intelligenz Workshop (KI'03), Preference learning: Models, Methods, Applications*, 2003.

[FM03b] R. Fullér and P. Majlender. On obtaining minimal variability owa operator weights. *Fuzzy Sets and Systems*, 136(2):203–215, 2003.

[FS92] B. R. Farr and R. D. Shachter. Representation of preferences in decision-support systems. *Computers and Biomedical Research*, 25(4):324–335, Aug 1992.

[FY98] D. Filev and R. R. Yager. On the issue of obtaining owa operator weights. *Fuzzy Sets and Systems*, 94:145–273, Mar 1998.

[FY02] J. Fodor and R. R. Yager. *Fuzzy Set-Theoretic Operators and Quantifiers*, volume Fundamentals of Fuzzy Sets of *The Handbooks of Fuzzy Sets Series*, chapter Fuzzy Sets, pages 125–194. Kluwer Academic Publishers, 2002.

[GBNP96] R. L. Grossman, H. Bodek, D. Northcutt, and V. Poor. Data mining and tree-based optimization. In *Knowledge Discovery and Data Mining*, pages 323–326, 1996.

[Gid85] B. Gidas. Non-stationary markov chains and convergence of the annealing algorithm. *Journal of Statistical Physics*, 39(1/2):73–131, Apr 1985.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and intractability; a guide to the theory of NP-completeness*. W.H. Freeman, 1979.

[GL79] G. V. Gens and E. V. Levner. Computational complexity of approximation algorithms for combinatorial problems. In *Symposium on Mathematical Foundations of Computer Science (MFCS)*, 1979.

[Gog65] J. A. Goguen. The logic of inexact concepts. *Synthese*, 19(3-4):325–373, 1965.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[Gol91] D. E. Goldberg. Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems*, 5:139–167, 1991.

[GOY02] M. Grabisch, S. Orlowski, and R. Yager. Fuzzy aggregation of numerical preferences. In R. Slowinski, editor, *Decision Making*, volume Fuzzy Sets in Decision Analysis Operations Research and Statistics of *The Handbooks of Fuzzy Sets Series*, chapter Decision Making, pages 31–68. Kluwer Academic Publishers, 2002.

[GS88] P. Gärdenfors and N. E. Sahlin, editors. *Decision, Probability and Utility*. Cambridge University Press, 1988.

[Haj88] B. Hajek. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13(2):311–329, 1988.

[Ham78] H. Hamacher. *Über logische Aggregationen nicht-binär explizierter Entscheidungskriterien; Ein axiomatischer Beitrag zur normativen Entscheidungstheorie*. Rita G. Fischer Verlag, Frankfurt, 1978.

[Hau99] W. Hauke. Using yager's t-norms for aggregation of fuzzy intervals. *Fuzzy Sets and Systems*, 101:59–65, 1999.

[HBC93] Y. Hayashi, J. J. Buckley, and E. Czogala. Fuzzy neural network with fuzzy signals and weights. *Int. Journal of Intelligent Systems*, 8(4):527–537, 1993.

[HF03] E. Hüllermeier and J. Fürnkranz. Preference learning: Models, methods, applications. Technical report, Österreichisches Forschungsinstitut für Artificial Intelligence, Wien, May 2003.

[HH97] V. Ha and P. Haddawy. Problem-focused incremental elicitation of multi-attribute utility models. In *Proc. of the 13th Conf. on Uncertainty in ArtificialIntelligence (UAI'97)*, pages 215–222, San Francisco, 1997. Morgan Kaufmann Publishers.

[HH99] V. Ha and P. Haddawy. A hybrid approach to reasoning with partially elicited preference models. In *Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence (UAI'99)*, pages 263–270, Stockholm, Sweden, 1999.

[HH03] V. Ha and P. Haddawy. Similarity of personal preferences: Theoretical foundations and empirical analysis. *Artificial Intelligence*, 146(2):149–173, 2003.

[HHB92] D. Heckerman, E. Horviz, and B.Nathwani. Towards normative expert systems: Part I, the pathfinder project. *Methods for Information in Medicine*, 31:90–105, 1992.

[HHB95] J. C. Hornberger, H. Habraken, and D. A. Bloch. Minimum data needed on patient preferences for accurate, efficient medical decision making. *Medical Care*, 33(3):297–310, May 1995.

[HHR$^+$03] P. Haddawy, V. Ha, A. Restificar, B. Geisler, and J. Miyamoto. Preference elicitation via theory refinement. *Journal of Machine Learning Research*, 4:317–337, 2003.

[HMS02] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. The MIT Press, 2002.

[Hof89] R. R. Hoffman. A survey of methods for eliciting the knowledge of experts. *ACM SIGART Bulletin*, pages 19–27, 1989.

[How66] R. A. Howard. Information value theory. *IEEE Trans. on Systems, Man, and Cybernetics*, 2(1):22–27, Aug 1966.

[How68] R. A. Howard. *The foundations of decision analysis*, volume 4. Prentice Hall, 1968.

[How77] R. A. Howard. *Risk preference*. Decision Analysis Group, SRI Int., Menlo Park, California, 1977.

[HS02] B. Hudson and T. Sandholm. Effectiveness of preference elicitation in combinatorial auctions. In *Revised Papers from the Workshop on Agent Mediated Electronic Commerce on Agent-Mediated Electronic Commerce IV, Designing Mechanisms and Systems (AAMAS'02)*, pages 69–86, London, UK, 2002. Springer-Verlag.

[IOT92] H. Ishibuchi, H. Okada, and H. Tabaka. Learning of neural networks from fuzzy input and fuzzy output data. *Journal of Japan Society for Fuzzy Theory and Systems*, 4(5):996–997, 1992.

[IS85] K. Ishii and M. Sugeno. A model of human evaluation process using fuzzy measure. *Int. Journal of Man-Machine Studies*, 22(1):19–38, 1985.

[ITS00] M. Inuiguchi, T. Tanino, and M. Sakawa. Membership function elicitation in possibilistic programming problems - a survey of recent developments. *Fuzzy Sets and Systems*, 111:29–45(17), Apr 2000.

[JAMS89] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation. part I, graph partitioning. *Operations Research*, 37(6):865–892, 1989.

[JAMS91] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation; part II, graph coloring and number partitioning. *Operations Research*, 39(3):378–406, May 1991.

[Jan93] J. R. Jang. Anfis: Adaptive-network-based fuzzy inference system. *IEEE Trans. on Systems, Man, and Cybernetics*, 23:665–684, 1993.

[JK02] W. Jang and C. M. Klein. Minimizing the expected number of tardy jobs when processing times are normally distributed. *Operations Research Letters*, 30(2):100–106, 2002.

[Joh02] D. S. Johnson. A theoretician's guide to the experimental analysis of algorithms. In *Data Structures, Near Neighbor Searches, and Methodology: 5th and 6th DIMACS Implementation Challenges*, pages 215–250. American Mathematical Society, Sep 2002.

[JSW98] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[KB74] M. Kochen and A. N. Badre. On the precision of adjectives which denote fuzzy sets. *Journal of Cybernetics*, 4:49–59, 1974.

[KD91] J. D. Kelly and L. Davis. A hybrid genetic algorithm for classification. In *Int. Joint Conf.s on Artificial Intelligence*, pages 645–650, 1991.

[KGK95] R. Kruse, J. Gebhardt, and F. Klawonn. *Fuzzy System*. Teubner, Stuttgart, Germany, 2nd edition, 1995.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220, 4598:671–680, 1983.

[KHS01] I. Kwee, M. Hutter, and J. Schmidhuber. Gradient-based reinforcement planning in policy-search methods. *Proc. of the 5th European Workshop on Reinforcement Learning (EWRL-5)*, 27:27–29, Oct 2001.

[Kle87] J. P. C. Kleijnen. *Statistical Tools for Simulation Practitioners*. Marcel Dekker, New York, 1987.

[Kle01] J. Kleijnen. Experimental designs for sensitivity analysis of simulation models. In A. W. Heemink, editor, *Proc. of the Federation of European Simulation Societies 2001*, Jun 2001.

[KM05] E. Kolman and M. Margaliot. Are artificial neural networks white boxes? *IEEE Trans. on Neural Networks*, 16(4):844–852, Jul 2005.

[KMP02] E. Klement, R. Mesiar, and E. Pap. *Triangular Norms*, volume 8 of *Trends in Logic*. Kluwer Academic Publishers, 2002.

[Koz92] J. R. Koza. The genetic programming paradigm: Genetically breeding populations of computer programs to solve problems. In B. Souček, editor, *Dynamic, Genetic, and Chaotic Programming*, pages 203–321. John Wiley, New York, 1992.

[KPP04] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer-Verlag, 2004.

[KR76] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley, New York, 1976.

[KW97] N. Kasabov and M. Watts. Genetic algorithms for structural optimisation, dynamic adaptation and automated design of fuzzy neural networks. In *IEEE: The Int. Conf. on Neural Networks (ICNN'97)*, 1997.

[KY95] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic, Theory and Applications*. Prentice Hall, Englewood Cliffs, 1995.

[LA87] P. J. M. Laarhoven and E. H. L. Aarts, editors. *Simulated annealing: theory and applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[Lak73] G. Lakoff. Hedges: A study in meaning criteria and the logic of fuzzy concepts. *Journal of Philosophical Logic*, 2(4):458–508, 1973.

[LCK08] K. C. Lee, H. R. Cho, and J. S. Kim. An expert system using an extended and/or graph. *Knowledge-Based Systems*, 21(1):38–51, Feb 2008.

[LHL97] G. Linden, S. Hanks, and N. Lesh. Interactive assessment of user preference models: The automated travel assistant. In A. Jameson, C. Paris, and C. Tasso, editors, *Proc. of the 6th Int. Conf. on User Modeling (UM'97)*, volume 383, pages 67–78, Wien, Jun 1997. Springer.

[LJS03] X. Luo, N. R. Jennings, and N. R. Shadbolt. Acquiring tradeoff preferences for automated negotiations: A case study. In P. Faratin, D. C. Parkes, J. A. Rodríguez-Aguilar, and W. E. Walsh, editors, *Agent-Mediated E-Commerce (AMEC)*, volume 3048 of *Lecture Notes in Computer Science*, pages 37–55. Springer, 2003.

[LNS02] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. DACE - a matlab kriging toolbox. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 2002.

[LR57] R. Luce and H. Raiffa. *Games and Decisions*. Wiley, New York, 1957.

[Man07] C. J. Mantas. A generic fuzzy aggregation operator: rules extraction from and insertion into artificial neural networks. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, Jul 2007.

[MCB79] M. McKay, W. Conover, and R. Beckman. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245, 1979.

[Men95] J. M. Mendel. Fuzzy logic systems for engineering: A tutorial. *Proc. of the IEEE*, 83(3):345–377, 1995.

[Mit97] T. M. Mitchell. *Machine Learning*. Mc Graw Hill, 1997.

[MJ02] J. M. Mendel and R. I. B. John. Type-2 fuzzy sets made simple. *IEEETFS: IEEE Trans. on Fuzzy Systems*, 10(2):117–127, 2002.

[Mor94] S. Morris. Revising knowledge: A hierarchical approach. In R. Fagin, editor, *Proc. of the 5th Conf. Theoretical Aspects of Reasoning about Knowledge (TARK'94)*, pages 160–174, San Francisco, 1994. Morgan Kaufmann.

[Mor02] B. Moret. Towards a discipline of experimental algorithmics. In M. H. Goldwasser, D. S. Johnson, and C. C. McGeoch, editors, *Data Structures, Near Neighbor Searches, and Methodology: 5th and 6th DIMACS Implementation Challenges*, volume 59, pages 197–213, Providence, 2002. American Mathematical Society.

[MR00] J. Marichal and M. Roubens. Determination of weights of interacting criteria from a reference data. *European Journal of Operational Research*, 124(3):641–650, 2000.

[MR05] P. Meyer and M. Roubens. *Choice, Ranking and Sorting in Fuzzy Multiple Criteria Decision Aid*, volume 78 of *In Operations Research & Management Science*, chapter Multiple Criteria Decision Analysis: State of the Art Surveys, pages 471–506. Springer New York, 2005.

[MS89] T. Murofushi and M. Sugeno. An interpretation of fuzzy measure and the Choquet integral as an integral with respect to a fuzzy measure. *Fuzzy Sets and Systems*, 29(2):201–227, 1989.

[MT90] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, New York, 1990.

[NH02] C. Nguyen and V. Huynh. An algebraic approach to linguistic hedges in zadeh's fuzzy logic. *Fuzzy Sets and Systems*, 129(2):229–254, 2002.

[Nil71] N. J. Nilsson. *Problem-Solving Methods in Artificial Intelligence*. Computer Science Series. McGraw-Hill, 1971.

[NK02] H. T. Nguyen and V. Kreinovich. *Methodelogy of fuzzy control: an introduction*, volume Fuzzy Systems Modeling and Control of *The Handbooks of Fuzzy Sets Series*, pages 19–62. Kluwer Academic Publishers, Boston/London/Dordrecht, 2002.

[NKaRK97] D. Nauck, F. Klawonn, and a. R. Kruse. *Foundations of Neuro-Fuzzy Systems*. Wiley, Chichester, 1997.

[NM53] J. V. Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. John Wiley and Sons, 1953.

[NS01] A. Nikov and S. Stoeva. Quick fuzzy backpropagation algorithm. *Neural Networks*, 14(2):231–244, 2001.

[Ovc02] S. Ovchinnikov. *An introduction to fuzzy relations*, volume Fundamentals of Fuzzy Sets of *The Handbooks of Fuzzy Sets Series*, chapter Fuzzy Relations, pages 233–259. Kluwer Academic Publishers, 2002.

[PP03] V. Peneva and I. Popchev. Properties ofthe aggregation operators related with fuzzy relations. *Fuzzy Sets and Systems*, 139:615–633, 2003.

[PR02] P. Perny and M. Roubens. *Fuzzy Preference Modeling*, volume Fuzzy Sets in Decision Analysis Operations Research and Statistics of *The Handbooks of Fuzzy Sets Series*, chapter Decision Making, pages 1–30. Kluwer Academic Publishers, 2002.

[PSZ95] P. M. Pardalos, Y. Siskos, and C. Zopounidis. *Advances in Multicriteria Analysis*. Kluwer Academic Publishers, Dordrecht, 1995.

[Pyz03] T. Pyzdek. *Quality Engineering Handbook*. CRC Press, 2003.

[RB94] M. Riedmiller and H. Braun. Rprop - description and implementation details. Technical report, University of Karlsruhe, 1994.

[Rec73] I. Rechenberg. *Evolutionsstrategie*, volume 15 of *problemata*. Friedrich Frommann Verlag, Stuttgart, 1973.

[Ren05] J. D. M. Rennie. Loss functions for preference levels: Regression with discrete ordered labels. In *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI), Multidisciplinary Workshop on Advances in Preference Handling*, pages 180–186, 2005.

[RHW86a] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and colleagues, editors, *Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations*. MIT Press, 1986.

[RHW86b] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagation errors. *Nature*, 323(99):533–536, 1986.

[Rot06] F. Rothlauf. *Representations for Genetic and Evolutionary Algorithms*. Springer Verlag, 2nd edition, 2006.

[Rou97] M. Roubens. Fuzzy sets and decision analysis. *Fuzzy Sets and Systems*, 90:199–206, 1997.

[Rud93] G. Rudolph. Massively parallel simulated annealing and its relation to evolutionary algorithms. *Evolutionary Computation*, 1(4):361–383, 1993.

[Saa74] T. L. Saaty. Measuring the fuzziness of sets. *Journal of Cybernetics*, 4:53–61, 1974.

[Saa94] T. L. Saaty. *Fundamentals of Decision Making and Priority Theory with the AHP*. RWS Publications, Pittsburgh, PA, USA, 1994.

[Sar95] D. Sarkar. Methods to speed up error back-propagation learning algorithm. *ACM Computing Surveys*, 27(4):519–544, 1995.

[Sav72] L. Savage. *The Foundations of Statistics*. Dover, New York, 2nd edition, 1972.

[Sch93] H. P. Schwefel. *Evolution and Optimum Seeking: The 6th Generation*. John Wiley & Sons, Inc., NY, USA, 1993.

[Sch97] M. Schonlau. *Computer experiments and global optimization*. PhD thesis, University of Waterloo, Waterloo, Canada, 1997.

[Ser94] M. Seraphin. *Neuronale Netze & Fuzzy-Logik*. Franzis-Verlag, München, 1994.

[SF07] P. Sevastjanov and P. Figat. Aggregation of aggregating modes in mcdm: Synthesis of type 2 and level 2 fuzzy sets. *The Int. Journal of Management Science*, 35:505–523, 2007.

[Skl03] D. Sklansky. *Hold'em Poker: A Complete Guide to Playing the Game*. Two Plus Two, 2003.

[SPR04] M. Serrurier, H. Prade, and G. Richard. A simulated annealing framework for ILP. *Proc. of the 14th Int. Conf. on inductive logic programming*, 1:289–304, 2004.

[SR95] H. Schwefel and G. Rudolph. Contemporary evolution strategies. In *European Conf. on Artificial Life*, pages 893–907, 1995.

[Sug74] M. Sugeno. *Theory of fuzzy integrals and its applications*. PhD thesis, Tokyo Institute of Technology, Tokyo, Japan, 1974.

[SW78] C. P. Shapiro and R. L. Wardrop. The bayes sequential procedure for estimating the arrival rate of a poisson process. *Journal of the American Statistical Association*, 73(363):597–601, 1978.

[SWMW89] J. Sacks, W. Welch, T. Mitchell, and H. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4:409–435, 1989.

[SWN03] T. J. Santner, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments*. Springer Verlag, New York, 2003.

[TF02] K. Tachibana and T. Furuhashi. A structure identification method of submodels for hierarchical fuzzy modeling using the multiple objective genetic algorithm. *Int. Journal of Intelligent Systems*, 17:495–513, 2002.

[Tor99] V. Torra. On the learning of weights in some aggregation operators. *Mathware and Soft Computing*, 6:249–265, 1999.

[Tor04] V. Torra. Owa operators in data modeling and reidentification. *IEEE Trans. of fuzzy systems*, 12(5):652–660, 2004.

[WCH$^+$95] K. Woods, D. Cook, L. Hal, K. Bowyer, and L. Stark. Learning membership functions in a function-based object recognition system. *Journal of Artificial Intelligence Research*, 3:187–222, 1995.

[WCM06] A. Waldock, B. Carse, and C. Melhuish. Hierarchical fuzzy rule based systems using an information theoretic approach. *Soft Computing*, 10(10):867–879, 2006.

[WCW$^+$08] W. Waegemana, J. Cottync, B. Wynsa, L. Boullarta, B. D. Baetsb, L. V. Langenhoved, and J. Detandc. Classifying carpets based on laser scanner data. *Engineering Applications of Artificial Intelligence*, 21:907–918, 2008.

[Wei02] K. Weicker. *Evolutionäre Algorithmen*. Teubner, Stuttgart, 2002.

[Wel85] M. P. Wellman. Reasoning about preference models. Technical Report MIT/LCS/TR-340, Massachusetts Institute of Technology, May 1985.

[WLW$^+$00] Z. Wang, K. Leung, M. L. Wong, J. Fang, and K. Xu. Nonlinear non-negative multiregressions based on choquet integrals. *Int. Journal of Approximate Reasoning*, 25(2):71–87, 2000.

[Won80] C. K. Wong. Minimizing expected head movement in one-dimensional and two-dimensional mass storage systems. *ACM Computing Surveys*, 12(2):167–178, 1980.

[Yag88] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans. on Systems, Man and Cybernetics*, 18(1):183–190, Jan 1988.

[Yag93] R. R. Yager. On a hierarchical structure for fuzzy modeling and control. *IEEE Trans. on Systems, Man, and Cybernetics*, 23(4):1189–1197, 1993.

[Yag00] R. R. Yager. A hierarchical document retrieval language. *Information Retrieval*, 3(4):357–377, 2000.

[YHF08] Y. Yi, E. Hüllermeier, and T. Fober. Fuzzy operator trees for modeling rating functions. Submitted to Int. Journal of Computational Intelligence and Applications, 2008.

[YL96] X. Yao and Y. Liu. Fast evolutionary programming. In *Evolutionary Programming*, pages 451–460, 1996.

[Zad65] L. A. Zadeh. Fuzzy sets. *Information and Controls*, 8:338–353, 1965.

[Zad71] L. A. Zadeh. Similarity relations and fuzzy orderings. *Information Sciences*, 3:177–200, 1971.

[Zad72] L. A. Zadeh. A fuzzy-set-theoretical interpretation of linguistic hedges. *Journal of Cybernetics*, 2:4–34, 1972.

[Zad73] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. on Systems, Man, and Cybernetics*, 3:28–44, 1973.

[Zae05] K. Zaeper. Intelligente methoden zur unterstützung der qualitätsbewertung in der robotergestützten messtechnik. Master's thesis, Philipps-Universität Marburg, 2005.

[ZH94] S. Zeng and Y. He. Learning and tuning fuzzy logic controllers through genetic algorithm. In *IEEE Int. Conf. on Neural Networks*, volume III, pages 1632–1637, Orlando, FL, USA, Jun 1994. IEEE press.

[ZZ80] H. J. Zimmermann and P. Zysno. Latent connectives in human decision making. *Fuzzy Sets and Systems*, 4:37–51, 1980.

Acknowledgements

I would like to express my gratitude to my supervisor Professor Eyke Hüllermeier for his excellent guidance through my PhD studies. Being a member of his research group *Knowledge Engineering & Bioinformatics*, I benefited from precise advice in many respects. I really appreciate the patience, support and encouragement he gave me in the last four years.

I am very grateful to the members of out *KEBI* working group, in particular Thomas Fober, Marco Mernberger, Jens Christian Hühn, Florian Finkernagel, Weiwei Chang and Stijn Vanderlooy for the helpful discussions and the careful proofreading of parts of the thesis.

Most of all, I want to thank my wife Siming for her loving support during the process of writing this thesis. She always tried to make a comfortable atmosphere for me.

Finally, I would like to express my deepest gratitude and appreciate to my parents in China.

Curriculum Vitae

Contact Information

	Yu Yi
	Am Richtsberg 88
	35039 Marburg, Germany
Email:	*yi@mathematik.uni-marburg.de*

Personal Information

Birthday:	24.06.1976
Birthplace:	Changsha, China
Citizenship:	China

Education

2006 – 2008	PhD student at University of Marburg, Germany
2005 – 2006	Master of Science at University of Magdeburg, Germany
2003 – 2004	Exchange student at University of Marburg, Germany
1995 – 2002	Studies of Computer Science at University Yanshan, China

Employment History

2006 – 2008	Research assistant, cooperation project “Industrielle Erforschung von Grundlagen und die Entwicklung

	und Umsetzung von Algorithmen in Software für ein Mess-System" between the Battenberg Robotic company and Prof. Eyke Hüllermeier University of Marburg, Germany
2005 – 2006	Internship at University of Magdeburg, Germany
1999 – 2002	Assistant teacher at University Yanshan, China

Publications

- Yu Yi, Eyke Hüllermeier, Thomas Fober. Fuzzy Operator Trees for Modeling Rating Functions. Submitted to *Int. Journal of Computational Intelligence and Applications*.

- Eyke Hüllermeier, Yu Yi. In Defense of Fuzzy Association Analysis. *IEEE Transactions on Systems, Man, and Cybernetics*. Part B, 37(4) : 1039 – 1043. 2007.

- Yu Yi, Eyke Hüllermeier. Learning Complexity-Bounded Rule-Based Classifiers by Combining Association Analysis and Genetic Algorithms. *Proceedings Eusflat–2005, 4th International Conference of the European Society for Fuzzy Logic and Technology*. September 2005. Barcelone.

- Yu Yi. Publication of the enterprise relational database with XML. *Computer Engineering*. 132-134. July 2002. Shanghai, China.

- Yu Yi. OLAP Implement Base On MDX. *The* 18-*th National Conference on Databases*. 213-215. August 2001. Qianghuang Dao, China.