2013-01-01

Extracting Fuzzy Measures from Sample Data: Optimization Algorithms and Applications

Xiaojing Wang
University of Texas at El Paso, xwang@utep.edu

EXTRACTING FUZZY MEASURES FROM SAMPLE DATA:

OPTIMIZATION ALGORITHMS AND APPLICATIONS

XIAOJING WANG

Department of Computer Science

APPROVED:

Martine Ceberio, Chair, Ph.D.

Vladik Kreinovich, Ph.D.

Heidi Taboada Jimenez, Ph.D.

Benjamin C. Flores, Ph.D.
Dean of the Graduate School

To my parents, Yushu Wang and Feng Xu,

my husband, Dong, and my beloved sons, Nathan and Chris,

for their endless love, support and encouragement

EXTRACTING FUZZY MEASURES FROM SAMPLE DATA: OPTIMIZATION ALGORITHMS AND APPLICATIONS

by

XIAOJING WANG

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

May 2013

Acknowledgements

First of all, I would like to express my deepest appreciation and gratitude to my advisor, Dr. Martine Ceberio of the Computer Science Department at the University of Texas at El Paso (UTEP), for her numerous guidance, enduring patience and encouragement during the doctoral research. Without her support and supervision, this dissertation could never have been finished. I sincerely appreciate her trust in me and her countless help through the entire dissertation preparation and writing process.

I would also like to thank my other committee members, Dr. Vladik Kreinovich of the Computer Science Department and Dr. Heidi Taboada Jimenez of the Department of Industrial, Manufacturing & Systems Engineering, for their advice, support and guidance through the proposal and the completion of this dissertation. Their valuable suggestions and comments have been very helpful to the completion of this work.

I would like to thank Dr. François Modave of the Paul L. Foster School of Medicine at Texas Tech University Health Sciences Center, for his encouragement on pursuing the doctoral degree and his guidance and help on identifying the research topic.

I am grateful to all professors and staff of the Computer Science Department for their help and support. I greatly acknowledge Angel F. Garcia Contreras, Christian Del Hoyo, and Luis C. Gutierrez from Constraint Research and Reading Group (CR2G) for their help on testing algorithms to complete this research.

Special thanks go to Jeremy Cummins in Youngstown State University for his help on initializing the algorithm.

My deepest appreciation and gratitude are also extended to all my colleagues in the Ph.D. study, all my friends and everyone who has helped directly or indirectly during my study.

Last but not least, I would like to thank my parents, Yushu Wang and Feng Xu, for their love and encouragement throughout my life. Thank you to my husband Dong, for his love,

support and understanding. I truly appreciate all that he has done for me over the years. I could not have reached here without his help and support. To my lovely sons, Nathan and Chris, I would like to express my thanks for always giving me unlimited happiness and pleasure.

Abstract

In Multi-Criteria Decision Making (MCDM), decisions are based on several criteria that are usually conflicting and non-homogenously satisfied. We use non-additive (fuzzy) measures combined with the Choquet integral to solve MCDM problems, for they allow to model and aggregate the levels of satisfaction of the several criteria of the problem at hand by considering the relationship between these criteria. In practice, it is difficult to identify such fuzzy measures. An automated process is then necessary and can be used when sample data is available. Several optimization approaches have been used to extract fuzzy measures from sample data, for example, genetic algorithms, gradient descent algorithms, and so on. We propose to automatically model experts' decision process and extract the fuzzy measure corresponding to their reasoning process; to do this, we use samples of the target experts' decision as seed data to automatically extract the fuzzy measure corresponding to the experts' decision process. In particular, we propose several approaches to extract fuzzy measures from sample data, including the Bees algorithm, an adaptive hybrid algorithm that combines the Bees algorithm and an interval constraint solver, and a speculative algorithm. We also apply the proposed approaches to the real data to show the applicability of the algorithms. The real applications include software quality assessment (SQA) and student risk level prediction. Our experimental results show that we are able to improve some of the results obtained through previous approaches, e.g., through machine learning techniques for software quality assessment problem and Cox proportional hazards (PH) regression model for student risk prediction problem.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

1.1.1 Multi-Criteria Decision Making

People make decisions every day. Some decisions are very simple; for example, where to eat lunch? what to eat? Such simple decisions usually do not take time to be made. Some decisions, however, are more complicated and may take longer to be made; for example, which car to purchase? which stock to invest in? All decisions, even more so the complicated ones, are made based on a list of alternatives and several criteria. In complicated decisions, criteria are usually conflicting (making the decisions harder). To make decisions, there usually are a list of alternatives and a list of criteria to be considered.

Example 1. *For instance, a breast cancer patient needs to decide on her treatment plan. The treatment options suggested by doctors include surgery that removes the cancer from the breast, then perhaps radiation therapy that uses x-rays or other types of radiation to kill cancer cells, and/or chemotherapy that uses drugs to stop the growth of cancer cells [27]. After talking to doctors, the patient decides on the treatment based on the stage of the cancer, the type of breast cancer, her age, general health, the benefit and the side effect of each treatment, etc [27]. In this example, the patient is the decision-maker: each treatment option is an alternative, and the stage of the cancer, the type of breast cancer, her age, general health, the benefits and side effects of each treatment are the criteria that will inform her decision (treatment choice).*

This kind of complex decision process is called **Multi-Criteria Decision Making (MCDM)**. MCDM is an important part of modern decision theory, defined as the "study of methods and procedures by which concerns about multiple conflicting criteria can be formally incorporated into the management planning process", by the International Society on Multiple Criteria Decision Making [46]. The research and the number of publications related to MCDM have drastically increased over the last ten years [62], and MCDM has been widely applied, especially in the following twelve key areas:

- environment management;
- water management;
- business and financial management;
- transportation and logistics;
- manufacturing and assembly;
- energy management;
- agricultural and forestry management;
- managerial and strategic planning;
- project management and evaluation;
- social service; and
- military service [62].

1.1.2 MCDM Method – Additive or Non-Additive?

Weighted Sum Method

In general, an MCDM problem consists of a set of alternatives, where each alternative is defined by a set of criteria that represent the characteristics of the alternative and are used to measure the alternative. When a decision based on several criteria is not critical, decision makers mentally "average/sort" the satisfaction levels of each criterion they have to consider. Usually, the satisfaction level of each criterion is subject to the decision maker's personal preference and some criteria might "weigh" more than others.

In MCDM, even when faced with the same set of alternatives and criteria, different decision makers may make different decisions based on their preferences. *Referring back to our earlier example, breast cancer treatment plans vary for different patients even if they are at the same stage of cancer, have the same type of breast cancer, same age, and similar health condition.*

Once criteria and satisfaction levels are identified (either implicitly – in the case of a daily decision – or through more sophisticated techniques in the case of critical decisions), the next task is to "combine" the satisfaction levels of each criterion into a global value, which is expected to match the decision maker's partial preferences. There exist many methods to attain such a global satisfaction level: one of the most widely used methods is the weighted-sum method: the overall score of an alternative is computed by aggregating the values of satisfaction along with weights on each criterion, where each weight indicates how important the corresponding criterion is when compared to the entire set of criteria. The weights assigned to the different criteria form what is called an "additive measure".

Example 1 (cont'd). *As the example we showed in 1.1.1, let us consider that the patient has an early-stage breast cancer and she is around 65 years old with good health. She has two*

surgery options (alternatives) to choose from: lumpectomy that removes only the cancerous area of the breast and some surrounding normal tissue, and mastectomy that removes the entire breast and usually some underarm lymph nodes. Lumpectomy is less invasive and removes the least amount of breast tissue. However, there is a chance that not all cancer cells will be removed, so radiation therapy is required. Mastectomy may cause psychological issues, however radiation therapy is usually not required [33]. The patient can base her decision on the following criteria: intactness of body, chance of recurrence of cancer, and requirement of radiation therapy. We show in Table 1.1 examples of what her satisfaction levels could be per criterion. This patient's objective is to live longer (regardless of what she has to undergo): as a result, less chance of recurrence is her most important criterion. Considering all criteria, mastectomy is the treatment that best matches her preference.

Table 1.1: MCDM Example – Breast Cancer Surgery

Alternative	Intactness of body	Chance of recurrence	Requirement of Radiation therapy	Overall Score
Lumpectomy	0.8	0.2	0.1	0.3
Mastectomy	0.1	0.8	0.8	0.66
Weight	0.2	0.6	0.2	

Example 2. *Let us look at another, more mundane, example of a decision-making process, in which we use a weighted-sum approach. A family wants to purchase a car and narrowed its search down to 4 candidates, as shown in Table 1.2. The family's selection criteria are security rating, price, city mileage (mpg), and acceleration (sec). Since different criteria have their own units, we transform them into a unique satisfaction level to make them easy to be compared. After the consideration, the family decides that the priorities of the importance of attributes are: security rating, followed by price, followed by acceleration, followed by city mileage. Security rating is the most important criterion for the family and*

it has the largest weight. For security rating, the family's satisfaction level over alternative cars is ordered as A_1, A_2, A_3, A_4, from the most to least preference. However, the family has a different satisfaction level with other criteria. For example, their satisfaction level for city mileage, the least important criterion, is A_3, followed by A_4, followed by A_1, followed by A_2. Taking into account all criteria, the overall score for each alternative is calculated by summing up the values of satisfaction and the weights for each criterion. As a result, A_2 has the largest overall score, and therefore, the family's most favorite option that matches all criteria is A_2, and their overall rating over all alternatives is A_2, A_4, A_1, A_3, from most to least preference.

Table 1.2: MCDM Example – Family Car Purchase

Alternative	Security Rating	Price	City Mileage	Acceleration	Overall Score
A_1	0.8	0.4	0.3	0.5	0.59
A_2	0.7	0.9	0.1	0.4	0.63
A_3	0.6	0.1	0.8	0.3	0.435
A_4	0.5	0.8	0.7	0.6	0.615
Weight	0.45	0.25	0.1	0.2	
Score of Alternative 1 = 0.45 × 0.8 + 0.25 × 0.4 + 0.1 × 0.3 + 0.2 × 0.5 = 0.59					

Example 3. *Let us consider another example of military performance evaluation. The criteria used to evaluate officers include professionalism, job knowledge, physical fitness, communication skill, and so on [25]. It turns out that an officer with outstanding technical ability but very bad communication skills is completely unsuitable for military service and should be put in "bad" class because with "bad" communication skill, s/he may not follow orders, or not report on what s/he is doing.*

When we use the weighted-sum method, we calculate the overall score for each alternative by using the satisfaction levels and the weights associated to each criterion only. We

assume that criteria are independent: we ignore the interaction among criteria. According to the weighted-sum method, an officer with bad communication skills but all other abilities outstanding may be selected over an officer with good communication skills whose other abilities are good but not outstanding – which goes contrary to our intent. In this example, the interaction among criteria cannot be ignored when making decisions, otherwise the decision would be inadequate.

Similar interaction between criteria needs to be taken into account in many real settings [7]. We observe that the criteria in the previous examples were conflicting: Which car to purchase that is energy efficient with better acceleration time? Which stock to invest in that has high return with low risk? Which treatment to choose that incurs less risk of recurrence with more intactness of the body? Sometimes, the criteria are not only conflicting but also strongly dependent. For example, the age of the breast cancer patient strongly affects the treatment benefit and the side effects.

Quality and price are also usually two conflicting and correlated criteria: although high-quality low-price items are sought, high quality usually goes with high price. Assume that a choice has to be made among four alternatives A, B, C, D with the following values of criteria {Quality, Price}: A = {high, low}, B = {high, high}, C = {low, low}, D = {low, high}. It is straightforward to decide that A is more preferable than D because A is better than D for both criteria. However, it is hard to decide the preference over B and C because neither of them satisfies both criteria, but only one criterion. There are three possible cases [21]:

- B and C are considered as bad as D, since neither of them satisfies both criteria. In this case, both criteria have to be satisfied and the importance of the set of criteria is larger than the importance of each individual criterion. The interaction between two criteria is positive.

- B and C are considered as good as A. In this case, it is enough to satisfy only

one criterion, which means the importance of the set of the criteria is the same as the importance of each individual criterion. The interaction between two criteria is negative.

- Since B and C satisfy only one criterion, they are better than D which satisfies none of the criteria and worse than A which satisfies both criteria. In other words, the importance of the set of the criteria is the "same" as the sum of the importance of each individual criterion, that is, the two criteria are independent and there is no interaction between them.

As a result, when considering the interaction between criteria (positive, negative, or independent), the decision may change. This situation is an illustration of what makes multi-criteria decision making hard to process and what makes weighted-sum approaches non-satisfactory.

A seemingly natural idea is to replace linear combination of criteria with quadratic combinations, if needed, by cubic combinations, etc. However, in some cases, these combinations also do not lead to relevant solutions [40]. As mentioned in [40], a more adequate approach is to use non-additive (fuzzy) measures.

Fuzzy Measures

If two criteria interact negatively, it means that both criteria express, in effect, the same attribute. As a result, when considering the set consisting of these two criteria, we should assign to this set the same weight as to each of these criteria and not double the weight as we would in a weighted-sum approach. In general, weights associated to different sets should differ from the sum of the weights associated to individual criteria. In mathematics, non-additive functions that assign values to sets are known as non-additive (fuzzy) measures. It is therefore reasonable to describe the dependence between different criteria by using an appropriate non-additive (**fuzzy**) **measure**.

Similarly to additive approaches, aggregation operators are used to calculate the overall

score of an alternative by using non-additive (fuzzy) measures and satisfaction levels. The integrals with respect to fuzzy measures are called fuzzy integral [18]. Two most well-known fuzzy integrals which are widely used in many applications are the Choquet integral and the Sugeno integral. The Choquet integral is an extension of the ordinary integral and the most natural fuzzy integral [44]. The Sugeno integral is similar to the Choquet integral and differs in the use of operators: the Choquet integral uses sum and product while the Sugeno integral uses maximum and minimum [11]. When the measure is additive, the Choquet integral coincides with the classic ordinary integral but this is not the case for the Sugeno integral restricted to additive measures [41]. For applications, the Choquet integral is better suited for multi-criteria decision making, and the Sugeno integral is better suited for possibility-like fuzzy measures, e.g., decision making under uncertainty [41]. In this study, we focus on the Choquet integral since we are interested in MCDM problems.

Given a set of alternatives to decide on and a set of criteria to describe them, the overall score of each alternative – that takes into account the dependence of criteria – is obtained by calculating the Choquet integral over a fuzzy measure defined on the given set of criteria. However, to make this happen, a fuzzy measure needs to be determined: it can either be identified by a decision maker/expert or by an automated system that extracts it from sample decision data. Since human expertise might not always be available and getting accurate fuzzy values (even from an expert) might be tedious [36], we focus in this dissertation on **extracting fuzzy measures from sample data**.

1.1.3 Fuzzy Measure Extraction from Sample Decision Data

The sample data that we use is a set of the actual preferences and the given inputs. Our objective is to provide a model that explains these preferences, i.e., to produce a fuzzy measure for which the application of Choquet interval to the criteria satisfaction levels leads to the observed decisions.

This problem is tackled as an optimization problem – several optimization approaches have been used to extract fuzzy measures from sample data, such as gradient descent algo-

rithms [16], genetic algorithms [8, 65, 72], and the neural networks [64], more specifically, as a constrained optimization problem since the optimal solution must also satisfy the monotonicity constraints inherent to the fuzzy measure we aim at determining.

The main contributions of this research work are: 1). the design of constrained optimization solving techniques for fuzzy measure extraction that are global and fast; **2).** the application of the resulting products to real-world situations. We first proposed several approaches to extract fuzzy measures from sample data, including Bees algorithm, an adaptive hybrid algorithm that combines the Bees algorithm and an interval constraint solver, and a speculative algorithm. Although the Bees algorithm had been successfully used in many optimization problems, it hadn't been applied to fuzzy measure extraction and required a significant amount of tuning to attain reasonable performance. We adjusted the Bees algorithm to the needs of fuzzy measure extraction problems for the first time. The adaptive hybrid algorithm we then proposed combined the Bees algorithm and RealPaver – an interval constraint solver – to shrink/bisect the search space, and improved the overall performance of the Bees algorithm while providing guarantees that the search focused on relevant parts of the search space and discarded irrelevant ones. In contrast to existing algorithms that focus on search space and use random search to increase the possibility of getting global results, the speculative algorithm we propose makes optimistic assumptions about the optimum value of the objective function before actually arriving to it.

1.1.4 Applications

We applied the above approaches to real-world situations to show the usefulness, applicability, and efficiency of our algorithms. Our real-world applications include software quality assessment (SQA) and student risk-level prediction. Both software quality assessment and student risk level prediction are Multi-criteria Decision Making (MCDM) problem, and therefore, can be solved by using optimization techniques.

Software Quality Assessment (SQA)

Software product quality is important because software is present in every aspect of normal day-to-day life. Software problems such as server breakdowns, software crashes, and data leaks have become common occurrences. Pre-existing software problems do not stop software spending. Even though there are large amounts of money spent on developing software, the quality of software still remains somewhat a mystery. The obvious questions therefore are: what is software quality and how is it measured? In this study, we rely on existing research and metrics for defining software quality, which defined the quality factors and linked all of them to measurable metrics in object-oriented software [3]. In general, there are six quality factors in this model, including Reusability, Flexibility, Understandability, Functionality, Extendability and Effectiveness, and each quality factor is associated with some metrics, e.g. Reusability is related to Coupling (DCC), Cohesion (CAM), Messaging (CIS) and Design Size (DSC). The detail of this model that defines the quality factors and links them to QMOOD set of metrics is in Chapter 6. Once metrics are adopted, it is relatively easy to decide, for each metrics, which piece of software is better than the others based on the measure/satisfaction level related to the corresponding metrics. However, even so, the problem of assessing software quality based on several metrics remains since it constitutes a complex decision involving several criteria. Experts are used to assess software quality, but it is desired that this process be automated so as to ensure the consistency of the assessments as well as its timeliness and cost. Such a task has been tackled previously by Fuentes et al. [47] using a machine learning approach. We show that the software quality assessment problem can be viewed as a multi-criteria decision making problem and solved accordingly, and therefore, software product quality can be predicted by using fuzzy measures and Choquet integral.

Student Risk Level Prediction

Improving retention and graduation rate is an important goal in higher education. Students risk-level classification can help institution implement appropriate strategies to help at-risk students keep studying in college, and therefore, improve the retention and graduation rates. Techniques to predict the level of risk students face in college have been implemented in the past and are still currently in use. These techniques use a set of selected predicting factors. Students risk-level predictions can therefore be seen as a multi-criteria decision-making problem. The straightforward reason for that is the following: a student's risk level can be identified according to his/her academic performance (i.e., cumulative GPA of a student), which is related to a set of predictor variables (i.e., multiple criteria that affect student's performance). Existing students data can be used to find prediction patterns (fuzzy measures) between the academic performance (GPA) and a set of other predictor variables. As a result, these patterns can be used to predict future students' performance, and then to classify the students' risk level. Therefore, fuzzy measure extraction for MCDM can be used for students risk-level prediction and we showed that our proposed algorithms allowed to do so efficiently.

1.2 Dissertation outline and contributions

This dissertation consists of three main parts.

- **The first part (Chapter 2 and 3)** gives an overview of MCDM, fuzzy measures and integrals, constraint optimization problems, and fuzzy measure extraction (FME).
  - In **Chapter 2**, we give an overview of the MCDM problem, common methods to MCDM, and we then focus on our research direction – fuzzy measure. The goal of our research is to extract fuzzy measures from sample data.
  - In **Chapter 3**, we identify the problem we have to solve as we show that fuzzy measure extraction (FME) is a constraint optimization problem.

- **The second part (Chapter 4 and 5)** reviews the existing approaches and presents our approaches.
  - In **Chapter 4**, we provide an overview of existing optimization techniques, including both techniques that have been applied to fuzzy measure extraction and those have not been yet applied to this problem.
  - In **Chapter 5**, we present the details of our approaches and our contributions. We first apply the Bees algorithm to FME, and show that using the Bees algorithm for FME yields promising performance, with results of similar or better accuracy than those obtained using existing approaches. We then combine the Bees algorithm with a global solver to shrink the search space, improve the overall performance of the Bees algorithm, and finally provide guarantees that the search has focused on relevant parts of the search space and discarded irrelevant ones. We also propose a speculative algorithm that focuses on the value of the objective function instead of the search space. For this minimization problem, we always bet that the optimum value of the objective function lies in the lower part of the function's range, and focus on this part of objective's range. Besides, we use a complete interval solver to verify our speculations on the value of the objective function which guarantees the solutions we reach are global. In this chapter, we also describe and report experimental results for the above approaches applied to toy examples.
- **The third part (Chapter 6)** describes our experiments and results on the real-world applications.
  - In **Chapter 6**, we give an overview of the real-world applications: software quality assessment and student risk-level prediction, together with the experiment results.
- The conclusions and the future work are discussed in **Chapter 7**.

All the original contributions of the dissertation have been published (or are in preparation for publication). In particular, the work presented in Chapter 5 has been published in [9, 69, 71], and the work presented in Chapter 6 has been published in [66, 67, 68, 70].

Chapter 2

Background

Multi-criteria decision making (MCDM) is the making of decisions based on multiple, usually conflicting criteria. It is an important part of the modern decision theory, and has developed very quickly in the last decades [62]. Today, MCDM has been widely applied in many fields, including business and finance, engineering, environment management, social service and evaluation, and so on.

There are two types of MCDM problems: multi-attribute decision making (MADM) and multi-objective decision making (MODM). We restrict our study on MADM. There are a lot of MCDM methods. Some methods require no information from the decision maker; some methods depend on the decision maker's attitude, pessimistic or optimistic; some methods require attribute weights measured by ordinal or cardinal scale. In contrast to methods that assume that the criteria are independent, (i.e., the decision maker can assign preference scores for the alternatives on one criterion without knowing what the alternatives preference scores are on any other criteria [12]), methods based on fuzzy measures and integrals constitute one of the MCDM class of methods that takes into account the interaction among criteria. We are interested in fuzzy measures and integrals in this study.

In this chapter, we first describe the multi-criteria decision making (MCDM) problem in Section 2.1. In particular, we review the definition, the classification, and the mathematical model of MCDM and briefly describe the common MCDM methods. In Section 2.2 and 2.3, we focus on fuzzy measures and integrals and show how to determine fuzzy measures in real-world settings.

2.1 Multi-Criteria Decision Making (MCDM)

In general, Multi-criteria decision making is defined as a 3-tuple $(X, I, \succeq)$ problem, where: we are given:

- X: a set of alternatives;
- I: a set of criteria;

and we have to determine:

- $\succeq$: a preference relation on the set of alternatives.

Alternatives represent the potential course of action available to the decision maker. The set of alternatives can be finite or infinite. They are screened, selected, and ranked [75]. An alternative could be a car to be purchased (as the example shown in Table 1.2), or a treatment plan to be chosen (as the example shown in Table 1.1).

Each MCDM problem is associated with multiple criteria which represent the different characteristics of the alternatives. The criteria usually have different units of measurement. In the car-purchase example we showed in the previous chapter (Table 1.2), price is expressed in dollars, gas mileage is in miles per gallon, acceleration is in seconds, and the security rating can be expressed either in numeric rating, such as 1, 2, 3, $\cdots$, or in literal rating, e.g., good, normal, poor, and so on.

For each criterion, the decision maker has his or her "partial" preference on the set of alternatives. As the example showed in Table 1.2, the decision maker's preference over the set of alternative cars for the security rating is A_1, A_2, A_3, A_4, from the most to the least preferable alternative. It is very common that no single alternative is the "best" for every criterion. This is illustrated in the same example in Table 1.2 as the decision maker's preference for the city mileage is A_3, A_4, A_1, A_2, which is different from his preference for the security rating. The challenge of MCDM problem is to find a global preference over all criteria that agrees with all of the "partial" preferences.

2.1.1 Classification of MCDM

MCDM can be classified into two main categories: *Multi-Attribute Decision Making (MADM)* and *Multi-Objective Decision Making (MODM)*. MADM is the making of decisions on a list of limited number of pre-specified alternatives in terms of a finite set of criteria (or attributes) [29, 45]. Let us see the example of purchasing cars again. The decision of the "best" car to purchase is based on the customer's preference on several criteria (or attributes) over a list of alternatives. The set of criteria as shown in Table 1.2 includes security rating, price, city mileage (mpg), and acceleration (sec), and the customer may consider more criteria, such as resale price, maintainability, passenger space, and so on.

In contrast to the MADM problem, MODM aims to find the "best" design of alternatives that optimizes the goals of the decision maker. In MODM, the alternatives are evaluated by how close they satisfy the decision maker's goal, and the set of criteria may not be finite [29, 45]. For example, consider the case of the design of the car. The goal is to find the best combination of the criteria that generates the most powerful and safety car.

In this research we focus on MADM problems, that is, when decisions are made over a finite set of alternatives and are based on a finite set of n criteria. The objective of MCDM is to make sense of a preference relation $\succeq$ on the set of alternatives in order to determine the best alternative.

2.1.2 Mathematical Model – An MCDM example

Let us use an example to describe the mathematical model of MCDM problem: a customer who wants to buy a car with high quality and low cost has three candidate cars, as shown in Table 2.1. Quality and cost are two criteria of the alternatives that the customer has to decide upon, and the possible values of these two criteria are $\{\texttt{low}, \texttt{average}, \texttt{high}\}$ and $\{\texttt{low}, \texttt{average}, \texttt{high}\}$. Although the problem looks simple, the challenge lies in how to deal with the trade-off between quality and cost.

The set of alternatives X can be represented as a multidimensional space, where $X \subseteq$

Table 2.1: Example of a MCDM problem

Alternative	Quality	Cost
A_1: Car 1	Low	Low
A_2: Car 2	Average	Average
A_3: Car 3	High	High

$X_1 \times \cdots \times X_n$ for n criteria. Let us note that X is likely to be only a subset of $X_1 \times \cdots \times X_n$ as this Cartesian product is likely to define some alternatives that are not feasible. Each alternative $x \in X$ can then be described by the tuple of the values it takes in each of the criteria; in particular, $x \in X$ is described by $x = (x_1, \ldots, x_n)$ where $\forall i \in \{1, \ldots, n\}$, $x_i \in X_i$. In this example, there might not exist any alternative $x \in X$ such that $x = (\texttt{low}, \texttt{high})$.

Each criterion i, where $i \in I = \{1, \cdots, n\}$, is associated to a set X_i of possible values of the corresponding criterion. For each $i \in I$, there is a preference relation $\succeq_i$ on each space X_i, such that for $x_i, y_i \in X_i$, $x_i \succeq_i y_i$ means that x_i is preferred to y_i. Since criteria for a given problem are usually measured in a difference scale or in a different level, either numeric or linguistic level. We need to transform them into a common scale to be able to compare them. We assume that for each criterion i, there exists a real-valued function $u_i : X_i \to R$ such that for all x_i, $y_i \in X_i$:

$$x_i \succeq_i y_i \iff u_i(x_i) \geq u_i(y_i)$$

Function u_i is called the i-th monodimensional utility function [42] and scales the values of all criteria onto a common (real) scale.

The MCDM problem with n criteria and m alternatives could be represented in a matrix format:

$$\begin{pmatrix} x_{11} & \cdots & x_{1i} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{j1} & \cdots & x_{ji} & \cdots & x_{jn} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mi} & \cdots & x_{mn} \end{pmatrix}$$

where $\forall i \in \{1, \ldots, n\}$ and $\forall j \in \{1, \ldots, m\}$, x_{ji} of the matrix indicates the normalized satisfaction level of the jth alternative with respect to the ith criterion.

Preference relations on the set of alternatives are not as straightforward to define as they are on the sets of values of the criteria. Preferences $\succeq$ over X are usually obtained from combining the preferences at the level of the criteria; in other words, $\succeq$ is defined from the $\succeq_i$'s, $i \in \{1, \ldots, n\}$. In particular, the utility functions are aggregated into a "global" preference (or utility) value $u(x)$. The best alternative (or consequence) is the $x \in X$ that yields the highest value of u. The way utility functions are combined is referred to as the aggregation operator.

The aggregation process that combines all partial preferences $\succeq_i$ into a global preference $\succeq$ can be either compensatory or non-compensatory [4, 23, 29, 75]. Compensatory refers to the existence of trade-offs between high and low performance of attributes. If there are no trade-offs then the aggregation is non-compensatory, otherwise, it is compensatory.

2.2 Fuzzy Measures and Integrals

Fuzzy measures are non-additive measures. They can be used to represent the degree of interaction of each subset of criteria [8]. In what follows, we consider a finite set of criteria $I = \{1, \cdots, n\}$.

Definition 1 *Let I be a finite set and $\mathcal{P}(I)$ the power set of I. A* fuzzy measure *(or a* non-additive measure*) defined on I is a set function $\mu : \mathcal{P}(I) \to [0, 1]$ satisfying the following conditions:*

(1) $\mu(\emptyset) = 0$

(2) $\mu(I) = 1$

(3) if $X, Y \subseteq I$ *and* $X \subseteq Y$, *then* $\mu(X) \leq \mu(Y)$

In Multi-Criteria Decision Making, fuzzy measures are used to show the importance of each subset of criteria with respect to others and to represent how each subset of criteria interacts with others. Fuzzy measures are expensive to determine: for a set of n criteria, the corresponding fuzzy measure has $2^n - 2$ values (as there are 2^n subsets of I and the values for the empty set and the complete set are known).

In order to show how different fuzzy measures are from additive approaches, let us assume that C_1 and C_2 are disjoint subsets of criteria. Then, in general, the measure value of $(C_1 \cup C_2)$ is not the same as the sum of the individual measure values associated to C_1 and C_2, that is:

$$m(C_1 \cup C_2) \neq m(C_1) + m(C_2).$$

In particular, in the cases we consider, the total measure value $m(A) = 1$ associated to the set A of all the criteria is, in general, different from the sum of the measure values $m(\{1\}) + m(\{2\}) + \ldots + m(\{n\})$ associated to all criteria. This is possible with the use of fuzzy measures but not with any additive approach.

Let us consider x, an element of the set of alternatives X, defined by $(x_1, \ldots, x_n)$. Now, just as we needed to do so earlier, we still need to aggregate the scores $u_i(x_i)$ by using an appropriate fuzzy measure. We can no longer add these scores with the measure values $m(\{i\})$ corresponding to individual criteria, because the sum of these values is, in general, larger than 1, so we need to decrease the measure values assigned to different criteria.

One possible approach is to use the most optimistic combination; after transforming the utility functions in such a way that the ranges of all u_i's are the same, we sort the

values x_i in increasing order of their utility value $u_i(x_i)$: $x_1 \leq x_2 \leq \ldots \leq x_n$, and then assign as large a measure value as possible to x_i corresponding to larger (more optimistic) values $u_i(x_i)$. We start with the value x_n with the largest utility and we associate it with the full measure value $m(\{n\})$. For the next best value x_{n-1}, we select the largest measure value for which the group consisting of the two best criteria is assigned its largest possible measure value $m(\{n-1, n\})$. Since we already know the measure value $m(\{n\})$ assigned to the n^{th} criterion, we can thus find the measure value to be assigned to the next best criterion as the difference $m(\{n-1, n\}) - m(\{n\})$. Similarly, we find the measure value for the $(n-2)^{\text{nd}}$ criterion from the condition that the three best criteria gets assigned the largest possible measure value $m(\{n-2, n-1, n\})$. This means that we assign, to this criterion, the measure value equal to the difference $m(\{n-2, n-1, n\}) - m(\{n-1, n\})$. In general, to the i^{th} criterion, we assign the measure value $m(\{i, i+1, \ldots, n\}) - m(\{i, \ldots, n\})$. With these measure values, we get the following combination:

$$\sum_{i=1}^{n} u_i(x_i) \cdot (m(\{i, i+1, \ldots, n\}) - m(\{i+1, \ldots, n\})). \tag{2.1}$$

This combination is known as the Choquet integral. Choquet integrals with respect to fuzzy measures are actively used in Multi-Criteria Decision Making [19].

In practice, in addition to Choquet integral, Sugeno integral is another widely used integral on fuzzy measures. Although the Sugeno and the Choquet integrals are structurally similar, they are different in nature [17]: the Sugeno integral is based on non-linear operators and the Choquet integral is usually based on linear operators. The applications of Sugeno and Choquet integrals are also very different [42]: the Choquet integral is generally used in quantitative measurements, and a MCDM problem usually uses a Choquet integral as a representation function. In this research, we focus on the Choquet integral, which we formally define hereafter.

Definition 2 *Let μ be a fuzzy measure on a set I. The* Choquet integral *of a function*

$f : I \to R$ with respect to μ is defined by:

$$(C)\int_I f d\mu = \sum_{i=1}^{n}(f(\sigma(i)) - f(\sigma(i-1)))\mu(A_{(i)})$$

where σ is a permutation of the indices in order to have $f(\sigma(1)) \leq \cdots \leq f(\sigma(n))$, $A_{(i)} = \{\sigma(i), \ldots, \sigma(n)\}$ and $f(\sigma(0)) = 0$, by convention.

Note: the above formula is equivalent to that of Equation 2.1 where $f(\sigma(i)) = u_i(x_i)$ and $\mu(A_{(i)}) = m(\{i, i+1, \ldots, n\})$.

2.3 Determining Fuzzy Measures

In MCDM, we would expect the decision maker to be more than likely to give the values of the fuzzy measure, but in most circumstances this is not the case. Attempts to making fuzzy measure identification easier for the decision makers have been made in [7], [61].

- In [7], the authors attempt to make this task easier by only requiring the decision maker to give an interval of importance for each interaction.
- In [61], the author suggests a diamond pair-wise comparison, where the decision maker only must identify the interaction of 2 criteria using a labeled diamond. From there, the algorithm evaluates the values of the numeric weights.
- In [61], the author discusses user specified weights mixed with an interaction index denoted λ or ξ. This algorithm is applied using an online aggregation application [60].

However, in most cases, the decision maker either does not understand the interactions well enough to provide a good value of each fuzzy measure, or does not have constant access to an expert who may give all values of the fuzzy measures. In addition, since there are 2^n values of a fuzzy measure for a problem with n criteria expert identification: it would be too time consuming anyway to be practical [16]. This is where fuzzy measure extraction

comes into play.

Summary. In this chapter, we briefly reviewed the MCDM problem, a decision making process based on multiple, usually conflicting criteria. In general, the MCDM problem consists in finding the most preferable alternative (decision) that best matches the decision maker's preference over the set of the criteria. There are many ways to aggregate the decision maker's partial preference on each criterion, and we are interested in fuzzy measure since it takes into account the interactions among criteria and is therefore more suitable in practice. The fuzzy measure is used to represent not only the importance of each criterion but also the importance of each subset of criteria. We use the Choquet integral with respect to a fuzzy measure to represent the satisfaction level of each alternative. However, in practice, it is difficult to rely on the decision maker and/or the expert to provide the accurate fuzzy measure to calculate the satisfaction level. In what follows, we focus on the problem of extracting fuzzy measures from sample data.

Chapter 3

Fuzzy Measure Extraction (FME) and Problem Formulation

As we mentioned in previous chapters, there are several ways to determine fuzzy measures in the real MCDM settings. The values of a fuzzy measure can be identified either by a decision maker/expert or by an automated process that extracts fuzzy measures from sample (training) data. Since human expertise might not always be available and it is difficult for a decision maker or expert to provide the accurate importance value of the criterion or the coalition of the criteria, we focus on extracting fuzzy measures from sample decision data.

In this chapter, we describe the fuzzy measure extraction (FME) problem in Section 3.1 and then show in Sections 3.2 and 3.3 that FME is a constrained optimization problem.

3.1 Fuzzy Measure Extraction (FME)

In MCDM with fuzzy measures and integrals, for lack of an expert to provide all values of the needed fuzzy measure, we have to determine fuzzy measures automatically. One way to do this is to use available decision data to inform us of the preferences/decision-making process of the target expert(s): we use sample data from prior decisions. Extracting a fuzzy measure is performed starting from such seed data. Fuzzy measure extraction seeks to find the fuzzy measure that returns the closest value to the expert(s)'s decision values: the preference/decision values that constitute our seed data set.

Let us take a look at the following situation: We consider a MCDM problem over a set

X of alternatives, each with n criteria, and we assume that we are given a sample data set of size m. Then, this MCDM problem can be expressed as follows:

$$\begin{aligned} y_1 &= F(x_{11}, \cdots, x_{1i}, \cdots, x_{1n}); \\ &\cdots \\ y_j &= F(x_{j1}, \cdots, x_{ji}, \cdots, x_{jn}); \\ &\cdots \\ y_m &= F(x_{m1}, \cdots, x_{mi}, \cdots, x_{mn}). \end{aligned} \tag{3.1}$$

where x_{ji} indicates the normalized satisfaction level of the jth alternative with respect to the ith criterion, y_j is the overall performance rating of the jth alternative with respect to all criteria, and F is the function showing the relation between $(x_{j1}, \cdots, x_{ji}, \cdots, x_{jn})$ and y_j for each alternative j in X.

If we knew the fuzzy measure $\tilde{\mu}$ corresponding to our problem, we would be able to compute preference values $\tilde{y}_j$ as for alternative j in the set of our alternatives X using the Choquet integral:

$$\tilde{y}_j = (C)\int_I f d\tilde{\mu} = \sum_{i=1}^{n} (f(\sigma_j(i)) - f(\sigma_j(i-1))\tilde{\mu}(A_{(i,j)}),$$

where σ is a permutation of the indices in order to have $f(\sigma_j(1)) \leq \cdots \leq f(\sigma_j(n))$ for alternative j, $A_{(i,j)} = \{\sigma_j(i), \ldots, \sigma_j(n)\}$ and $f(\sigma_j(0)) = 0$. For example, for the problem with $n = 4$ criteria, if the normalized satisfaction levels for the jth alternative are $x_{j2} \leq x_{j4} \leq x_{j3} \leq x_{j1}$, then $f(\sigma_j(1)) = x_{j2}$, $f(\sigma_j(2)) = x_{j4}$, $f(\sigma_j(3)) = x_{j3}$, $f(\sigma_j(4)) = x_{j1}$, and $A_{(1,j)} = \{1, 2, 3, 4\}$, $A_{(2,j)} = \{1, 3, 4\}$, $A_{(3,j)} = \{1, 3\}$, $A_{(4,j)} = \{1\}$.

However, with the sample data we have access to, we only have access to the decision values of a subset of X. In order to compute preference values of other alternatives in X, we need to determine a fuzzy measure μ that characterizes the input-output system described in Equation 3.1. In general, $2^n - 2$ coefficient values of the fuzzy measure need to be determined.

3.2 Optimization

We determine μ such that the value $\tilde{y}_j$ corresponding to the computed Choquet integral is as close to the decision values y_j of the sample data as possible. We do so by considering a least square approach and, as a result, we aim at minimizing the following sum (and getting it as close to 0 as possible) [20]:

$$e = \sum_{j=0}^{m} (y_j - \tilde{y}_j)^2 \tag{3.2}$$

which can be rewritten as:

$$e = \sum_{j=0}^{m} \left(y_j - \sum_{i=1}^{n} (f(\sigma_j(i)) - f(\sigma_j(i-1)))\mu(A_{(i,j)}) \right)^2 \tag{3.3}$$

When $e = 0$, the identified fuzzy measure μ is the exact solution of the problem, i.e., the extracted fuzzy measure perfectly models the expert(s)'s decisions: this is the ideal case. In most cases, we put up with "only" reaching an approximate model, that is, with $e \neq 0$ but close to 0. The reason for such a weaker expectation is that the sample data might not be fully consistent with a fuzzy measure.

In any case, extracting a fuzzy measure is cast down to solving an optimization problem. Regardless of whether e is equal to 0 or not, the solution, or the approximate optimal solution is not necessarily unique. At least one of the solutions or the approximate optimal solutions can be obtained using optimization techniques.

In a word, the fuzzy measure extraction (FME) problem can be represented as an optimization problem which seeks to find fuzzy measures μ that minimize the following objective function:

$$\min e = \sum_{j=0}^{m} \left(y_j - \sum_{i=1}^{n} (f(\sigma_j(i)) - f(\sigma_j(i-1)))\mu(A_{(i,j)}) \right)^2 \tag{3.4}$$

where

- y_j is the decision value of the jth alternative in X;

- μ is a fuzzy measure defined on the power set $\mathcal{P}(I)$ of I; and
- $\sum_{i=1}^{n}(f(\sigma_j(i))-f(\sigma_j(i-1)))\mu(A_{(i,j)})$ is the Choquet integral with respect to the fuzzy measure μ. It is used to calculate the performance rating for the jth alternative.

The size of this optimization problem is the size of the fuzzy measure to be determined, i.e., $2^n - 2$. In known cases of using Choquet integral, n varies from 3 to 6, in which cases the size of the fuzzy measure varies from 6 to 62. In principle, the number n of attributes can be much larger than six.

3.3 Constrained Optimization and Problem Formulation

In this section, we show that the optimization problem described earlier is actually a constrained optimization problem since the values of the fuzzy measure μ that the minimization process seeks must satisfy monotonicity properties that characterize a fuzzy measure (as seen in Chapter 2 and Section 3.2).

Rather than minimizing an objective function over a given search space, the fuzzy measure extraction (FME) problem also targets constraints that need to be satisfied. In other word, a solution to such a problem is defined by any element of the search space, among only those that satisfy all constraints, that minimizes the objective function. What this means is that the search for the optimum is only conducted in the feasible space of the constraints (elements of the search space that meet the constraints): constraints are requirements that are imposed and that must be met; when a candidate meets the constraints it then can be considered a possible solution. The goal of constrained optimization is to find the most optimal solution that meets the constraints.

So, in the case of fuzzy measure extraction, fuzzy measures must be monotonic and must always take values between 0 and 1. These are considered the constraints. For the

problem with n criteria, the number of monotonicity constraints is $\sum_{k=1}^{n-2} \binom{n}{k} * (n-2)$.

The fuzzy measure extraction problem is an optimization problem subject to constraints (monotonicity), which can be reformulated as follows:

$$\min e = \sum_{j=0}^{m} \left(y_j - \sum_{i=1}^{n} (f(\sigma_j(i)) - f(\sigma_j(i-1)))\mu(A_{(i,j)}) \right)^2 \tag{3.5}$$

subject to:

- if $S_1, S_2 \in \mathcal{P}(I)$, and $S_1 \subseteq S_2$, then $\mu(S_1) \leq \mu(S_2)$

- $0 \leq \mu(S) \leq 1, \forall S \in \mathcal{P}(I)$

where I is a set of criteria, and $\mathcal{P}(I)$ is the power set of I.

Summary. In this chapter, we formulated fuzzy measure extraction (FME) as a constrained optimization problem that seeks to find fuzzy measures that not only best match the sample decision values but also satisfy monotonicity constraints. In the next chapter, we will briefly review existing optimization techniques. Some of them have already been applied to fuzzy measure extraction (FME), and we will show how these algorithms work for FME and explain their limitations. In addition, there are optimization algorithms, which are efficient and convenient to use but have not been applied to FME yet. We will show that some of them that may apply to FME in the future.

Chapter 4

Optimization Techniques

In Chapter 3, we showed that fuzzy measure extraction (FME) can be expressed as a constrained optimization problem that looks for fuzzy measures which minimize the difference between existing seed decision data and corresponding decisions computed using a fuzzy measure and the Choquet integral. Many algorithms exist to solve optimization problems. Some of these optimization algorithms are based on numerical linear or nonlinear programming approaches, others are heuristic algorithms based on simulations, e.g., simulating natural evolution in genetic algorithms, simulating physical annealing process in simulated annealing, simulating the natural harmony creation process in harmony search. Some of them have already been proposed to extract fuzzy measures from sample data, such as genetic algorithm, gradient descent algorithm, and neural networks. We briefly go over the most significant ones in Section 4.1. Some common used heuristic optimization algorithms that have not yet been applied to FME are reviewed in Section 4.2.

4.1 Optimization Algorithms Used for FME

Several optimization algorithms have been used for fuzzy measure extraction, including genetic algorithms, gradient descent algorithm, and neural networks.

4.1.1 Gradient Descent Approach

Gradient descent algorithm, also known as steepest descent, is a general iterative method for solving the minimization problem $\min_{x \in \mathbf{R}^n} f(x)$. The method moves the point x_i toward x_{i+1} by following the direction of steepest descent, that is, the direction that x_i is moved

to decrease f the fastest. The gradient descent algorithm is straightforward and easy to implement. It usually converges to a local minimum and may take a long time to converge depending on the selection of the initial point [24].

In [16], Grabisch proposed a gradient descent algorithm with constraints for identifying fuzzy measures that takes advantage of the "lattice structure of the coefficients" of the fuzzy measure. Using this lattice structure, it is easier to find the "path" to be used to calculate the Choquet integral, where the "path" is the set of values for all the set $\mu(A_i)$ and the "path" used depends on the input values, e.g., for the problem with $n = 4$ criteria and for the jth alternative, if the permuted normalized value are: $f(\sigma(1, j)) = x_{j2}$, $f(\sigma(2, j)) = x_{j4}$, $f(\sigma(3, j)) = x_{j3}$, $f(\sigma(4, j)) = x_{j1}$, then coefficients on the path involved calculating the Choquet integral are: μ_1, μ_{13}, μ_{134}, μ_{1234}, where μ_{134} denotes $\mu(\{x_1, x_3, x_4\})$, as emphasized in Figure 4.1.

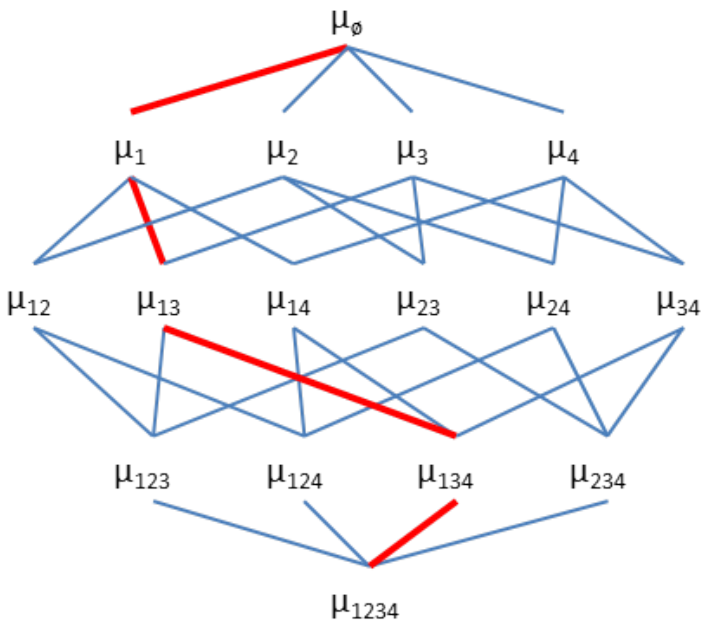


Figure 4.1: Lattice structure of a fuzzy measure (n=4) [16]

This algorithm starts from the uniform coefficient of the fuzzy measure μ and consists

of two steps. The first step modifies only the coefficients on the path used to compute the Choquet integral in order to decrease the error. The monotonicity has to be verified in this step. The second step modifies unmodified coefficients left in the first step by using the average value of their neighbors. There is no need to verify the monotonicity in the second step since it has been checked in the first step and the calculations used in the second step do not violate it. These two steps are repeated until a stopping criterion is satisfied.

This algorithm can reach a local optimum quickly and accurately. But the monotonicity constraints need to be verified at every iteration, which may take time and lead to backtracking a significant number of times. This algorithm was improved on in [1] by reducing the gradient magnitude of error surface. According to the experiments reported in [1], the modified gradient descent algorithm is noticeably more accurate, a little faster than the original gradient descent algorithm, and is 385 times faster than a genetic algorithm with similar or better accuracy (on the tested toy examples).

However, such algorithm often falls into a local optimum, and even if a global optimum is reached, there is no easy way to prove it.

4.1.2 Genetic Algorithms

A genetic algorithm is a heuristic optimization method inspired by natural evolution. The basic idea of the genetic algorithm is to simulate the processes in natural evolution, especially following the principles of survival of the genetically fittest. In general, genetic algorithms perform random searches within a defined search space to solve the problem.

Genetic algorithms start with a seed population that contains a set of individuals (solutions). Each individual is characterized by a sequence of genes that build a chromosome. The individual in the population is evaluated by a fitness function, which measures how well the individual performs/contributes to the solution of the problem. The individuals are then selected for mating to generate a new population according to their fitness. The more suitable individuals have more chances to be selected to reproduce the off-springs. Usually, the new population is generated through a three-step process: selection, crossover,

and mutation. This process is followed to make sure that the new generation not only inherits the parents' properties but also has some genetic diversity. The fitness of each individual in the population is calculated again, and if all individuals in the population are identical, the algorithm stops, meaning that the optimal solution is found. This is the ideal case, however, in most cases, the genetic algorithm is stopped after a predefined number of iterations. The individual(s) with the best fitness are the optimal solutions of the problem. The solutions are not restricted to be unique, and it is possible that several different individuals have the same best fitness.

Genetic algorithms have been used successfully to solve a number of optimization problems, including fuzzy measure extraction in [8, 65, 72]. In fuzzy measure extraction, the sequence of genes usually represents possible coefficient values of the fuzzy measure μ, and therefore, the individual represented by the chromosome is the fuzzy measure to be determined. The fitness function is usually defined by the objective function, *i.e.,* the difference between the expert's decision data and the predicted decision through the Choquet integral. As designed in the work references above, the initial population is usually randomly generated, and in the selection process the way the parents are chosen varies between each implementation of the algorithm. In some versions, the parents are chosen at random and then mate through a mix of the crossover, where part(s) of each parents are selected but remain in the same location on the off-spring and realignment, where part(s) of each parent are put in a new position in the offspring [65]. In others, the parents are chosen by fitness and then mate through crossover, which convex combines two fuzzy measures and guarantees that the new generated population follows the monotonicity of fuzzy measures but reduces the search space in each generation. Finally, the mutation is performed by changing some individuals randomly to keep a high diversity [8].

Such algorithms were tested on sample data and the results were compared with the gradient descent algorithm in [16]: the comparison showed that the genetic algorithm of [16] performed better than the compared genetic algorithm on the tested examples when noise was present, meaning that genetic algorithms were more suitable for practical situations

since noise is impossible to avoid [8].

Genetic algorithms show promise for solving fuzzy measure extraction problems, especially since they have been used successfully in the past. However, they also suffer from the risk that they fall into a local optimum. While mutation is included in the algorithm to try to avoid it, it does not totally prevent it (especially if there are local optima that are in distant locations but have values close to the global optimum).

4.1.3 Neural Networks

Neural networks (also called artificial neural networks) are parallel computing systems inspired by biological neural networks and try to simulate the human learning process. Neural networks can be used to solve a variety of problems including optimization problems. A neural network usually consists of a large number of artificial neurons with many interconnections and can be modeled as a weighted directed graph where artificial neurons are nodes and interconnections between neuron inputs and neuron outputs are directed edges with weights.

There are two main types of neural networks: feed forward neural networks (FFNNs) and recurrent neural networks (RNNs) [53]. In FFNN there are no feedback loops, and the information is passed in the forward direction. In RNN there are feedback loops. For instance, single layer feed forward networks consist of a single layer of computational nodes, where the inputs are directly connected to the outputs, via a series of weights. A more complicated neural network, called multi-layer network, is a neural network with one or more hidden layer. In a multi-layer network, the links between nodes can go directly from an input layer to an output layer through one or more hidden layer, which is the case of feed forward networks. It is also possible that the links can connect nodes in the same layer or go back to connect the nodes in the previous layers, which is the case of recurrent neural networks. In general, a neural network is trained with known samples of data. The network updates its weights from the given data trying to minimize some cost function of errors between the training data and its own output [53].

An algorithm using a neural network for fuzzy measure extraction was proposed in [64]. In this work by Wang, the relation between the input, output, and the fuzzy measure is described by a multi-layer feed forward neural network. The relation between the input and the fuzzy measure is nonlinear and complex, while the relation between the fuzzy measure and the output (Choquet integral) is linear. Then, a gradient-descent technique is used to train the neural network and to find an optimum (hopefully global) within the search space. Although the algorithm was successfully used to determine fuzzy measures, the authors realized that using neural network algorithm, the search easily falls into a local minimum [64].

4.1.4 Main limitations

Although previous attempts have been shown to extract fuzzy measures successfully from sample data, they share the following limitations.

(1) The returned solution (found minimum of the objective function) might just be a local minimum. There is no guarantee that it would be the global minimum at all.

(2) Uncertainty might be part of the model to solve. It is reasonable for experts to provide data in ranges instead of precise values. Using intervals allows taking uncertainty into account but none of the above techniques handled intervals/uncertainty.

(3) When dealing with problems defined on real numbers, the actual computations will round each real number to the most "relevant" floating-point number. Rounding errors can lead the returned result to be dramatically different from the original expected solution [55].

4.2 Other Optimization Algorithms

There are a lot of optimization algorithms that have not been applied to FME yet. In what follows, we briefly describe a few of them, focusing on evolutionary and local optimization algorithms rather than on global optimization algorithms as the size of the problems to be solved is deemed too large for global optimization algorithms (namely interval ones) to

handle them.

4.2.1 Bees Algorithms

The Bees optimization algorithm, proposed in [50], uses bees' natural food foraging habits as a model for the exploration of the search space. The Bees algorithm combines a local and "global" search that are both based on bees natural foraging habits.

The algorithm starts by placing "scout bees" in the search space randomly and calculating the fitness of sites visited by scout bees, called the patches of "nectar". The bees with the highest fitness are chosen as "selected bees". More bees are sent out to search in the neighborhood of the patches visited by "selected bees". When a new patch is found, its fitness is evaluated and compared against previously explored patches. If a local search "bee" finds a better patch of "nectar", the location from where it was dispatched is moved to the new location [50]. Meanwhile, a number of scout bees are kept searching in the search space for better patches of "nectar". This keeps the algorithm searching "globally". Together with global search performed by scout bees, the local search around the best patches is the key operation of the Bees algorithm [50].

It is believed that the best ranked "patch" when a stopping criterion is met is the optimal solution. In the work the authors of [50] did, applying the algorithm is, in most cases, faster than a number of other algorithms, and returned a solution within 0.1% of the perfect solution every time run. They believe that the new algorithm proposed is up to 207 times faster than a genetic algorithm. The algorithm also presents an improvement over the simplex method and the stochastic simulated annealing optimization procedure. The functionality of using real life modeled methods is also shown as the genetic algorithm and the Bees algorithm were the only algorithm tested to achieve success 100% of all runs. They also state that the Bees algorithm always finds the global optimum and ignores local optimum, see [50].

4.2.2 Harmony Search (HS)

Harmony search was first developed by Geem et al. in 2001 [15]. It is a music-based optimization algorithm, simulating the musical process of searching for a perfect state of harmony [15].

Harmony search starts by creating a harmony memory and randomly filling it with "pitches" played by each of the "instruments". Each harmony in the harmony memory represents one solution. Next, a new harmony is improvised based on three operations: random consideration, memory consideration, or pitch adjustment. In random consideration, the new harmony is improvised by randomly choosing pitches within the range of "instrument". In memory consideration, the pitches of the new harmony can be chosen from any pitches already stored in the harmony memory. In pitch adjustment, after the pitch is chosen from the harmony memory, it can be further adjusted to the neighboring pitch. The algorithm then evaluates the fitness of the new harmony. The new harmony is then placed in the proper position in the harmony memory, replacing the worst harmony, assuming its fitness is better than the worst harmony [15].

The harmony search has been successfully applied to many scientific and engineering problems. As authors mentioned in [15], comparing to other heuristic algorithms, such as genetic algorithms, harmony search considers all existing values in memory to generate the new generation (harmony) rather than consider only two parents as in the genetic algorithm. It also does not require setting the initial values of variables which makes the harmony search to be more flexible.

4.2.3 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) is a population-based stochastic optimization algorithm developed by Kennedy and Eberhart in 1995 [30]. It is inspired by the social behavior of bird flocking or fish schooling.

In particle swarm optimization (PSO), each solution is a "bird" in the flock or a "fish"

in the school and is described as a "particle". Similar to the genetic algorithm, PSO starts with a population of random solutions and searches for optima by updating generations. However, there's no evolution operators such as crossover and mutation in PSO to generate the offsprings, particles in the population follow their social behavior and move towards a destination [13]. To find a destination, each particle spread in the search space and move randomly and independently. They communicate together and identify the best location. Each particle then moves towards the best location and investigates the search space from its new local position. The process repeats until an "ideal" destination is found [13, 51].

In PSO, the way the particles moved is based on the particle's own best position, the best position of the local best particle(s), and/or the best position of the global best particle(s) [51]. The degree that all three movement criteria effect the movement of the particle depends on the specific algorithm. The authors of [51] found that algorithms that concentrate on the particles own best along with the best of the best local particle(s) to have the most success avoiding falling into local optimum.

4.2.4 Ant Colony Optimization (ACO) and Modified Ant Colony Optimization (MACO)

Ant colony optimization (ACO) is based on the behavior of ants seeking food with the shortest path between the nest and the food. Ants leave their nest and randomly search for food. When ants find the food, they carry the food and return back by following their pheromone trails and depositing more pheromone. Each ant that senses the pheromone follows the trail and leaves its pheromone. In general, the path with the highest density of pheromone is the shortest one since ants choose it to return back to the nest with food [13].

In [37], the authors propose a mix of the ant colony optimization algorithm and the genetic algorithm, called modified ant colony optimization (MACO).

MACO performs both local and "global" searches simultaneously. In the algorithm, first the problem space is split into a number of randomly distributed regions. The regions

are then ranked according to fitness. A local and "global" search is then begun. In the local search, ants begin by moving in a random direction from each region. If an improvement is found, the region's position is updated, a proportional amount of pheromone to the locations fitness is deposited, the region's age counter is reset to zero, and the ant will move the same direction the next iteration. If no improvement is found, the ant will move in a random direction during the next iteration. The distance the ant moves is proportional to the age of the region, maximum for age 0 and decreasing as the region gets older. The "global" search is done by essentially the same method as the Genetic Algorithm, "mating" two parents (in this case regions), and introducing mutations. The offspring then replaces the least fit region [37].

4.2.5 Simulated Annealing (SA)

Simulated Annealing, proposed by Kirkpatrick et al. in 1983, is a probabilistic method for finding the global optimum in a large search space [31]. It is often used when the search space is discrete [10]. This method works by simulating the physical annealing process. In metallurgy and material science, annealing is a heat treatment that cools a metal down slowly and freezes it into a defect-free crystal structure that represents the minimum energy state.

Simulated annealing (SA) method randomly searches through the problem space, looking for points with low energies. It starts from a random initial solution. In each iteration, the simulated annealing algorithm generates a new solution that takes values from the neighbors of the current solution. The new solution is compared with the current solution, and the better one is accepted and then defined as the current solution from which the search repeats until the stopping criteria are met.

The main idea of simulated annealing is it avoids local optima by accepting both better and worse solutions [10].

Because genetic algorithm easily falls into a trap and finds a local minimum, an adaptive hybrid algorithm (ARSAGA) that combines genetic algorithm with simulated annealing is

proposed to insure the solution quality and to improve the convergence speed [26]. The experimental results indicate that both the global searching ability and the convergence speed are significantly improved.

Summary. In this chapter, we went over some optimization algorithms that have been applied to fuzzy measure extraction (FME), such as, the gradient descent algorithm which is a traditional optimization algorithm and seeks to improve the solution in the neighborhood of the starting point, the genetic algorithm which is a heuristic algorithm and seeks to find the optimal fuzzy measures by mimicking the biological evolution process, the neural network which simulates human being's learning process to find the best solution. We also went over some heuristic optimization algorithms that have not been applied to FME yet, including the Bees algorithm, harmony search, particle swarm optimization, ant colony optimization, and simulated annealing.

Starting in the next chapter, we will focus on our contributions, starting with our contribution to algorithms for fuzzy measure extraction (FME) problem. In particular, we are interested in the Bees algorithm. It is a relative new algorithm that models the way bees find food and has not been applied to FME yet. We picked the Bees algorithm because it performs "global" search. The more important reason we chose the Bees algorithm is it is faster than other algorithms as we already described in 4.2.1. We will show how we adjusted the Bees algorithm and applied it to FME, and then combined the Bees algorithm with a global constraint solver to guarantee that the search is focused on relevant part of the search space. We also proposed a speculative algorithm that focuses on the objective function's range rather than on the search space.

Chapter 5

Contributed Optimization Algorithms and Results

In Chapter 4, we briefly reviewed some heuristic optimization algorithms that have not been applied to fuzzy measure extraction (FME). These heuristic methods mimic natural phenomena, such as physical annealing process in simulated annealing (SA), harmony creation process in harmony search (HS), bees food foraging habits in Bees algorithm, bird flocking or fish schooling in particle swarm optimization (PSO), ants seeking food behavior in ant colony optimization (ACO). Similar to genetic algorithm, these heuristic algorithms can also be applied to FME.

In this chapter, we first apply one of these algorithms, the Bees algorithm, to fuzzy measure extraction (FME) and solve this problem as a constrained optimization problem. We then propose an adaptive hybrid approach that combines the Bees algorithm with a global constraint solver to shrink the search space. We also propose a speculative algorithm that focuses on the value of objective function instead of the search space.

The Bees algorithm and the adaptive hybrid approach are presented in Section 5.1 and the speculative algorithm in Section 5.2.

5.1 Hybrid Optimization Techniques

5.1.1 Bees Algorithms

As described in Section 4.2.1, Bees algorithms have proved to be faster than genetic algorithms and have also presented improvements over some other optimization procedures. We believe that the Bees algorithm shows the most promise for use in creating a new algorithm for fuzzy measure extraction (FME) of the methods discussed, and therefore, we propose to apply the Bees algorithm to extract fuzzy measures from sample data.

The pseudocode and flowchart of the Bees algorithm are illustrated in Table 5.1 and Figure 5.1, respectively.

The Bees algorithm starts by placing a number of scout bees (n_{bees}) randomly in the search space, and the fitness of n_{bees} sites (solutions) visited by scout bees is then evaluated and ranked. Here, the fitness is defined by

$$e = \sum_{j=0}^{m} \left(y_j - \sum_{i=1}^{n} (f(\sigma_j(i)) - f(\sigma_j(i-1)))\mu(A_{(i,j)}) \right)^2 \tag{5.1}$$

as described in Chapter 3, where n is the number of criteria and m is the number of samples. In the case of FME, the smaller e is the better fitness. A number of best m_{sites} sites are then selected from n_{bees} for more bees to perform the local search. The local search size (sc) determines the search radius from the selected sites. Among m_{sites} selected sites, a number of best sites (es), called "elite sites", is then selected according to the fitness, i.e., the first es sites. A number of "elite bees" (n_1) are dispatched to look for a better site around the "elite sites" in random direction. After evaluating the fitness of the n_1 bees, only the bee with the best fitness is selected to be part of the next bee population. For the rest ($m_{sites} - es$) selected sites, called "other sites", a number of "other bees" ($n_2 < n_1$) are sent out to randomly search the neighborhood area of the "other sites" in hope of better sites, and the fittest bee among the n_2 of each such site is selected to be part of the next population. Since "elite sites" have better fitness than "other sites", "elite sites" are more

Table 5.1: Pseudocode of the Bees algorithm

Input: populationSize (n_{bees}), #selectedSites (m_{sites}), #eliteSites (es), localSearchSize (ngh), eliteBees (n_1), otherBees (n_2), localSearchSizeShrink (sc), precision (e_0), totalNumberOfIteration ($totalIter$)

Output: valueOfBestBees

```
1 BeesArray ← InitializePopulation(n_bees) // Bees are randomly generated
2 Evaluate(BeesArray)
3 Sort(BeesArray) // in increasing order
4 WHILE (numIter < totalIter & BeesArray[0] > e_0)
5         IF (numIter ≥ sc)
6               ngh ← ngh/10 // Shrink local search size ngh
7         ENDIF
8         FOR each i ∈ {1, ··· , m_sites} // we focus on each of the first m_sites Bees
9               IF (i < es) // Bee is at "elite site"
10                    EliteBeesArray ← CreateEliteBeesArray(n_1) // send n_1 Bees to the "elite site"
11                    LocalSearch(EliteBeesArray, ngh)
12                    Evaluate(EliteBeesArray)
13                    Sort(EliteBeesArray) // in increasing order
14                    IF (EliteBeesArray[0] < BeesArray[i])
15                        BeesArray[i]=EliteBeesArray[0]
16                    ENDIF
17              ELSE // Bee is at "other site"
18                    OtherBeesArray ← CreateOtherBeesArray(n_2) // n_2 < n_1
19                    LocalSearch(OtherBeesArray, ngh)
20                    Evaluate(OtherBeesArray)
21                    Sort(OtherBeesArray)
22                    IF (OtherBeesArray[0] < BeesArray[i])
23                        BeesArray[i]=OtherBeesArray[0]
24                    ENDIF
25        END
26        FOR each j ∈ {m_sites + 1, ··· , n_bees − m}
27              BeesArray[j] ← CreateRandomBees()
28        END
29 Evaluat(BeesArray)
30 Sort(BeesArray) // in increasing order
31 ENDWHILE
32 Return valueOfBestBees ← BeesArray[0]
```

promising than "other sites", and more bees are sent to "elite sites" ($n_1 > n_2$). This is made to enhance the chances that the optimal solution be found quickly and efficiently. The remaining bees in the population are randomly relocated in the search space looking for new potentially good sites: this keeps the algorithm searching globally. After some iteration, the local search size (sc) is shrunk to focus more on the location which is closer to the selected site. To further improve on the time to solution, when we evaluate the fitness of population, we add penalties to the evaluation values if the monotonicity of the fuzzy measure is violated: the penalty is large enough (penalty =10) to ensure that penalized sites do not have a chance to be selected. This way backtracking time is saved.

We expect that using the Bees algorithm to extract fuzzy measures from sample data will provide better performance than existing approaches without losing accuracy. We also expect that a closer-to-global optimum will be reached by using the Bees algorithm because of its "global" search.

5.1.2 Adaptive Hybrid Algorithm

Our second approach to fuzzy measure extraction aims at further enhancing the solving process, more specifically, to increase the chances to reach a global optimum. We also aim to allowing the possibility for more uncertainty in the model (when decisions are uncertain) and to avoid errors possibly caused by floating-point computations. To do so, we design a new algorithm that pairs our previous Bees algorithm for FME with an interval constraint solver, RealPaver [22]. RealPaver, as described in Table 5.2, is a complete, interval-based, continuous constraint solver for modeling and solving nonlinear system. In general, a constraint solving problem which includes a set of constraints and a bounded domain of variables is sent to RealPaver, and RealPaver returns the solution set that satisfies all the constraints using a "branch-and-prune" algorithm, which is an iterative method that shrinks and/or bisects the domains until reaching the precision [22]. Using RealPaver allows us to focus the Bees' search on relevant and smaller areas of the search space by pruning areas that do not match the currently found optimum. RealPaver is repeatedly requested

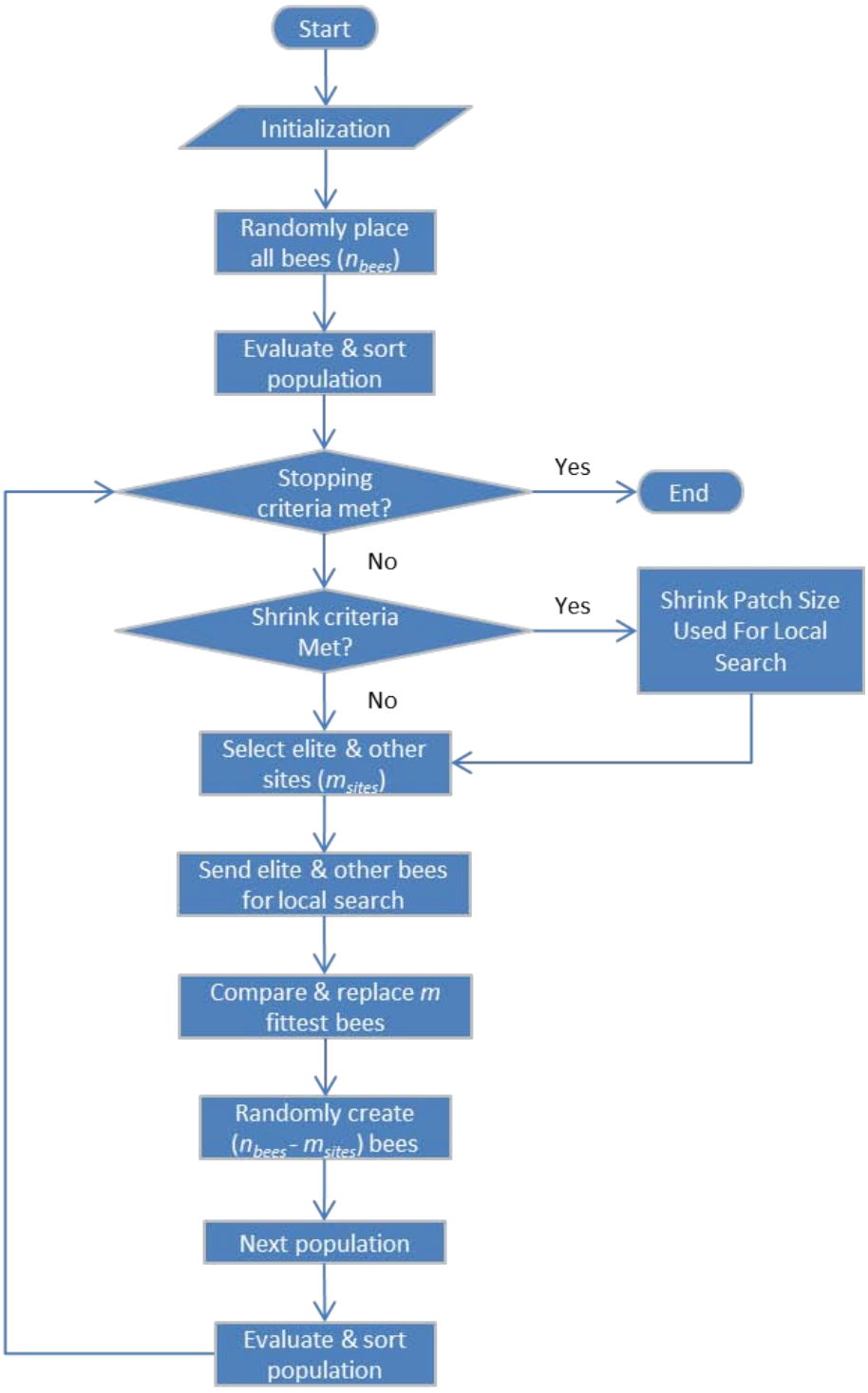


Figure 5.1: The Bees Algorithm

to solve the following continuous and non-linear constraint problems:

$$\boxed{\begin{array}{l} e = \sum_{A_i} \{\text{SQA}(A_i) - \text{Choquet}(\mu, \text{metrics}(A_i))\}^2 \leq f^* \\ \mu \text{ satisfies monotonicity constraints} \\ \mu \text{ is a fuzzy measure} \end{array}} \tag{5.2}$$

where f^* is the best value our Bees algorithm has been able to identify at the time RealPaver is solicited. RealPaver then helps discard parts of the search space that do not satisfy the above constraints. In particular, it focuses the search on areas of the search space that are more likely to yield a better optimum.

Figure 5.2 illustrates the pairing of the Bees algorithm with RealPaver: the Bees algorithm first performs an exploration of the search space and converges on some areas of it (as denoted by the black dots in the image on the left-hand side). From this exploration, the Bees algorithm is already able to provide an upper bound for the value of the objective function (which is, and which we call, f^*). A constraint solving problem of the form shown in Equation 5.2 is then sent to Realpaver (right hand-side of Fig. 5.2), which in turns will reduce the search space by shrinking it: i.e., non-feasible (w.r.t. considered constraints) parts of the search space that lie on the outskirt of it are discarded. The reduced search space is then sent back to the Bees algorithm for resuming of the former search and further exploration.

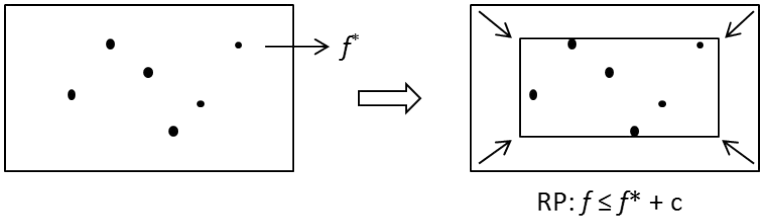


Figure 5.2: Using RealPaver to shrink the search space

Figure 5.3 illustrates a situation that can happen during the pairing of Bees and RealPaver:

Table 5.2: Real Paver

RealPaver is software for modeling and solving linear and non-linear systems that uses interval methods and constraint satisfaction techniques [22]. RealPaver takes constraint problems with real interval domains as input and returns outer approximations of the solution sets. Outer approximations are (a set of) subdomains whose union contains all solutions of the original problem. The output of RealPaver could then consist of one box (cartesian product of) subdomains or multiple boxes that contain all feasible solutions.

The main property of RealPaver is reliability as claimed in [22].
Property 1 (Reliability) *RealPaver computes a union of boxes that contains all the solutions of the constrained satisfaction problem. If no box is computed by RealPaver, then the constrained satisfaction problem has no solutions.*

RealPaver generates boxes from the initial box (search space) using an iterative method that implements a branch-and-prune algorithm. The pruning step aims at reducing a box by eliminating inconsistent values from domains. The branching step splits a box into a set of sub-boxes in order to separate the solutions. RealPaver guarantees that any boxes that are pruned do not contain any solutions.

although it is sought that RealPaver will shrink the search space, it might happen that it does not. This means that shrinking the search space (outside-in) would potentially remove an element of the search space that is seen as a feasible element for the constraints. As a result, in such a situation, since we still aim to provide some focus to the Bees search by narrowing the search space, we then split the search space and recall RealPaver of each of the two halves: if shrinking at the outskirt of the original search space was not possible, it might be possible to discard parts of the search space within the original space.

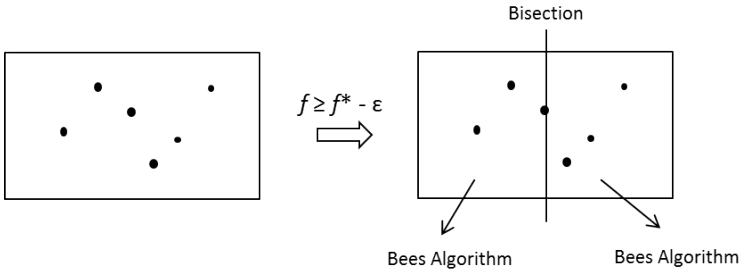


Figure 5.3: Using RealPaver to bisect the search space

The flow chart of the combination of the Bees algorithm and RealPaver is illustrated in Figure 5.4.

In order to account for situations as described above, and to explore best configurations, we considered three modes of interaction of Bees and RealPaver, which we called Hybrid1, Hybrid2, and Hybrid3. They are described below:

1. Hybrid1: both RealPaver and the Bees algorithm are called only once. RealPaver is called at the start of the process to shrink the search space based on the fuzzy measure constraints (monotonity and range), and it is followed by the Bees algorithm that then only considers the resulting search space for exploration.

2. Hybrid2: both RealPaver and the Bees algorithm are called multiple times. They alternate until a given accuracy of the solution is reached. No bisection is allowed (i.e., the scenario of Figure 5.3 is not allowed).

3. Hybrid3: this version is similar to Hybrid2 except that it allows bisection. In this case, RealPaver and our Bees algorithm are invoked alternatively until there is no more reduction of the search space: at this point, the search space is split and the search is resumed on each of the halves, as shown in Figure 5.3 – the no-bisection mode is also restored until the search is stalled again. The process ends when a given accuracy of the solution has been reached.

5.1.3 Experiments and Results

Table 5.3: Fuzzy measure to be identified ($n = 4$)

A	$\mu(A)$	**A**	$\mu(A)$	**A**	$\mu(A)$
$\{1\}$	0.1	$\{1,2\}$	0.3	$\{1,2,3\}$	0.5
$\{2\}$	0.2105	$\{1,3\}$	0.3235	$\{1,2,4\}$	0.8667
$\{3\}$	0.2353	$\{1,4\}$	0.7333	$\{1,3,4\}$	0.8824
$\{4\}$	0.6667	$\{2,3\}$	0.4211	$\{2,3,4\}$	0.9474
		$\{2,4\}$	0.8070		
		$\{3,4\}$	0.8235		

The goal of our experiments is to extract the fuzzy measure μ that is presented in Table 5.3. To test the performance of the Bees algorithm, we follow the same procedure as in [16] and [8], as described hereafter. The input-output system contains 81 sample data with $Y = f_c(X) + g$, where Y is the vector of the system outputs (experts' decisions), X is a 4-tuple input (x_1, x_2, x_3, x_4), where x_i is the experts' preference on ith criterion and $f_c(X)$ is the normalized satisfaction level, and g is a centered Gaussian noise. In the same manner as [16] and [8] did, we also check 5 different variances $\sigma^2 = 0.0$, 0.00096, 0.00125, 0.00625, and 0.0125, respectively. Our algorithm is executed on an Intel Xeon

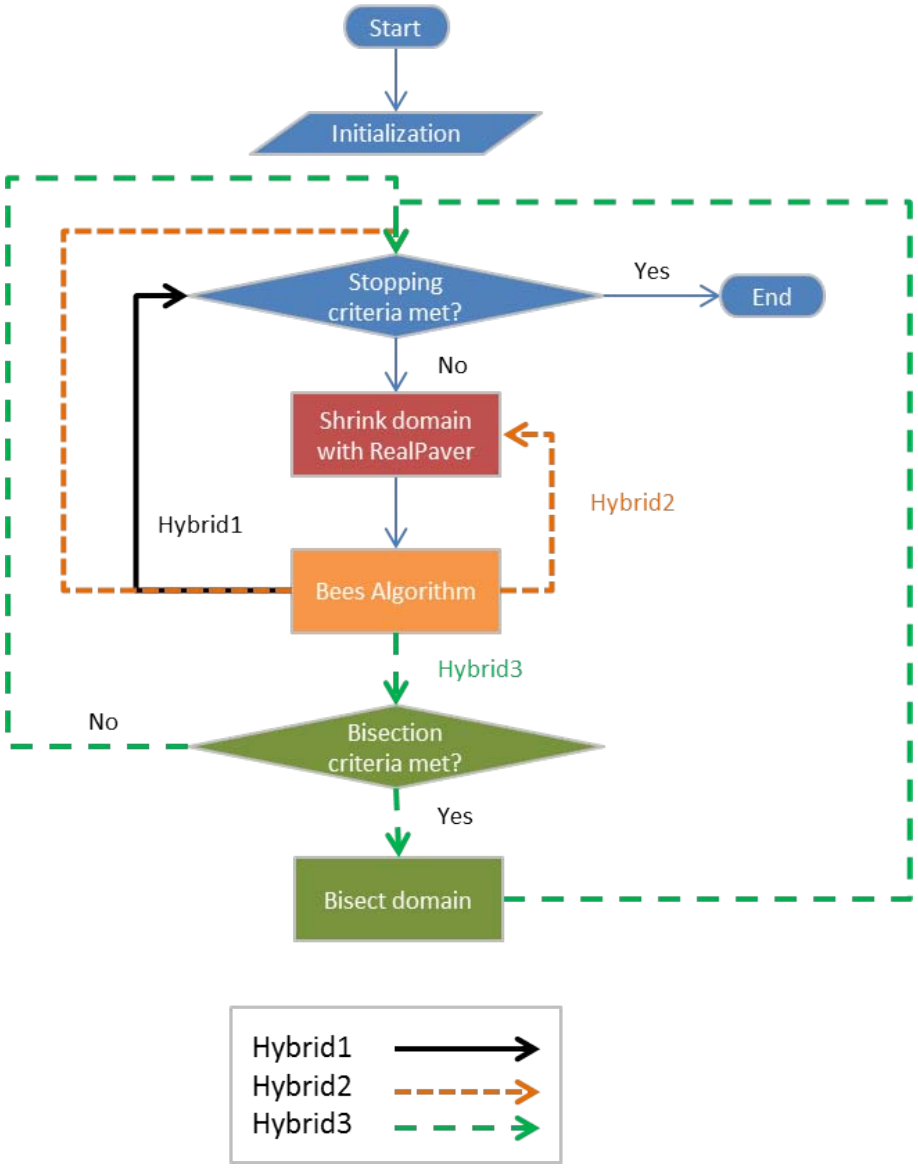


Figure 5.4: Combined Bees Algorithm & RealPaver (Simplified)

e5540 @2.53GHz machine. The experiments are run 10 times, and we calculate the average as the final result. The parameters used for the algorithm are presented in Table 5.4. In order to find the optimal solution (most fitting fuzzy measure), we are interested in the minimum mean square error E, where $E = e^2 = \frac{1}{81}\sum_{i=1}^{81}(y_i - \hat{y}_i)^2$.

Table 5.4: Parameters

Parameter	**Value**
Number of total iterations	100
Number of scout bees	50
Number of selected sites	30
Number of elite sites	10
Number of elite bees	100
Number of other bees	50
Initial patch size	0.001
Number of iterations before local search size shrink	10
Monotonicity enforced by penalty/fix	Penalty

Quality of Solutions

Table 5.5 shows our experimental results. We compare them with the results of other algorithms described in 4.1 in the same table. The results of other algorithms are taken from [16] and [8], respectively.

Table 5.5: Comparison of Bees algorithm and other algorithms

σ^2	Gradient Descent [16]		Genetic Algorithm [8]		Bees algorithm	
	no. iter	E	no. iter	E	no. iter	E
0.0	8	1.4E-7	1000	0.00141472	100	1.47E-06
0.00096	10	0.00083	1000	0.00147230	100	0.000509
0.00125	11	0.01080	1000	0.00141241	100	0.001106
0.00625	11	0.05300	1000	0.00183267	100	0.004566
0.01250	9	0.10540	1000	0.00241865	100	0.009878

We observe that when the variance of the Gaussian noise is increased, the mean square error is increased accordingly. However, the mean square error E is always smaller than the original variance σ^2 of the random noise, which means our algorithm does not generate negative influence to the data.

Compared to the gradient descent algorithm in [16], our algorithm performs much better when the variance is greater than zero. Comparing to the genetic algorithm in [8], our algorithm has better results when variance is small, but when the variance keeps increasing (≥ 0.00625), the results of our algorithm becomes worse.

The final obtained fuzzy measures for different noises are in Table 5.6. When $\sigma^2 = 0$, the obtained fuzzy measure values are very close to the fuzzy measures to be identified. With the increase of the noise, the differences also increase.

Table 5.7 shows our experimental results for 1000 iterations using hybrid approaches. We compare them with the results of the Bees algorithm in the same table. We observe that the mean square errors using hybrid approaches are slightly smaller than those obtained when using the Bees algorithm alone, which means that our hybrid approaches improve the quality of the extracted fuzzy measure. The final obtained fuzzy measures using Hybrid3 are reported in Table 5.8, and we can observe that they are very close to what we expected

Table 5.6: Obtained fuzzy measures using the Bees algorithm

μ	**0.0**	**0.00096**	**0.00125**	**0.00625**	**0.01250**
$\mu(\{1\})$	0.104380	0.080039	0.076429	0.058450	0.072574
$\mu(\{2\})$	0.208948	0.200362	0.272064	0.127179	0.235488
$\mu(\{3\})$	0.239601	0.246264	0.305229	0.408975	0.296089
$\mu(\{4\})$	0.666377	0.637343	0.717156	0.663812	0.671832
$\mu(\{1,2\})$	0.293485	0.347558	0.331216	0.337807	0.235542
$\mu(\{1,3\})$	0.321726	0.279074	0.340944	0.409823	0.457894
$\mu(\{1,4\})$	0.729298	0.759373	0.718045	0.664870	0.673014
$\mu(\{2,3\})$	0.420896	0.417667	0.421213	0.484186	0.296426
$\mu(\{2,4\})$	0.809941	0.816434	0.719934	0.858011	0.764084
$\mu(\{3,4\})$	0.820760	0.797741	0.809397	0.790631	0.763582
$\mu(\{1,2,3\})$	0.499038	0.473590	0.458572	0.491641	0.459702
$\mu(\{1,2,4\})$	0.864815	0.823745	0.881903	0.935657	0.873362
$\mu(\{1,3,4\})$	0.881175	0.881366	0.833616	0.842672	0.937774
$\mu(\{2,3,4\})$	0.946263	0.953104	0.943259	0.996405	0.893088

Table 5.7: Comparison of Bees algorithm and hybrid approaches(Total number of iteration = 1000)

σ^2	Bees Algorithm	Hybrid1	Hybrid1	Hybrid3
0.0	2.54E-7	1.38E-7	1.59E-7	2.56E-8
0.00096	0.0007004	0.0007006	0.0007004	0.0007004
0.00125	0.0012948	0.0012945	0.0012944	0.0012943
0.00625	0.0055944	0.0055943	0.0055945	0.0055943
0.01250	0.0143783	0.0143775	0.0143771	0.0143777

to get.

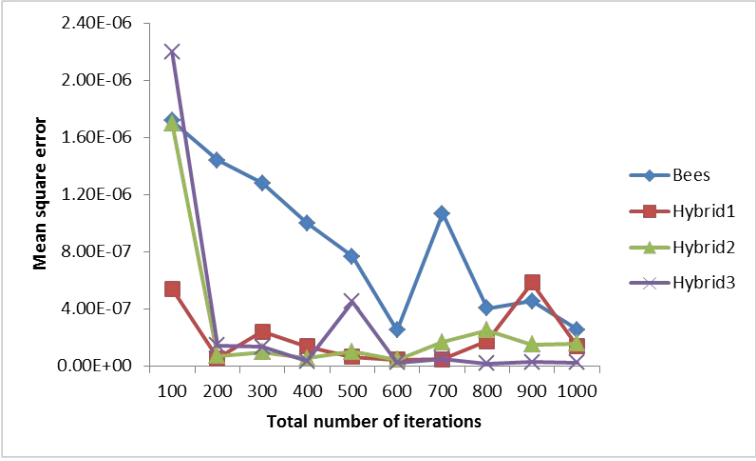


Figure 5.5: Convergence Based on Number of Iterations ($\sigma^2 = 0.0$)

Figures 5.5 to 5.9 present the comparison of the convergence diagram for samples with different noise. We observe that with the same number of iteration for the Bees algorithm (comparing Bees and Hybrid1), using RealPaver, even when it is allowed to shrink the search space only once, helps to improve the performance. However, with the same total number of iteration (iterations for each Bees call multiplies the number of Bees calls),

Table 5.8: Obtained fuzzy measures using Hybrid3

μ, σ^2	0.0	0.00096	0.00125	0.00625	0.0125
$\{1\}$	0.1000229	0.1056434	0.0821659	0.0462540	0.1380325
$\{2\}$	0.2105267	0.1843541	0.1763852	0.1611654	0.1295627
$\{3\}$	0.2353900	0.2242320	0.2466740	0.2772939	0.1619649
$\{4\}$	0.6665281	0.6694569	0.6777330	0.5052198	0.6673205
$\{1,2\}$	0.3000861	0.3130654	0.3115890	0.2889373	0.2776955
$\{1,3\}$	0.3232191	0.3143200	0.3844483	0.2815279	0.5913198
$\{1,4\}$	0.7332572	0.6781726	0.6827574	0.8223756	0.6845643
$\{2,3\}$	0.4211996	0.4566696	0.3972005	0.4599749	0.3210012
$\{2,4\}$	0.8068813	0.7907318	0.8086925	0.8476835	0.8375995
$\{3,4\}$	0.8234168	0.8587281	0.8081816	0.8437407	0.8271190
$\{1,2,3\}$	0.4999835	0.5025472	0.5008131	0.5744260	0.5913410
$\{1,2,4\}$	0.8666455	0.8929275	0.8087053	0.9582186	0.9380317
$\{1,3,4\}$	0.8824617	0.8755151	0.8770698	0.8771552	0.8783669
$\{1,3,4\}$	0.9478859	0.9702236	0.9741276	0.9379999	0.9900401

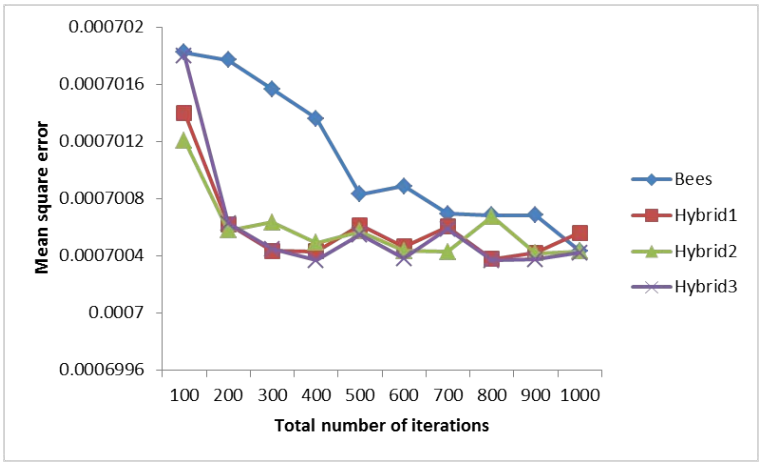


Figure 5.6: Convergence Based on Number of Iterations ($\sigma^2 = 0.00096$)

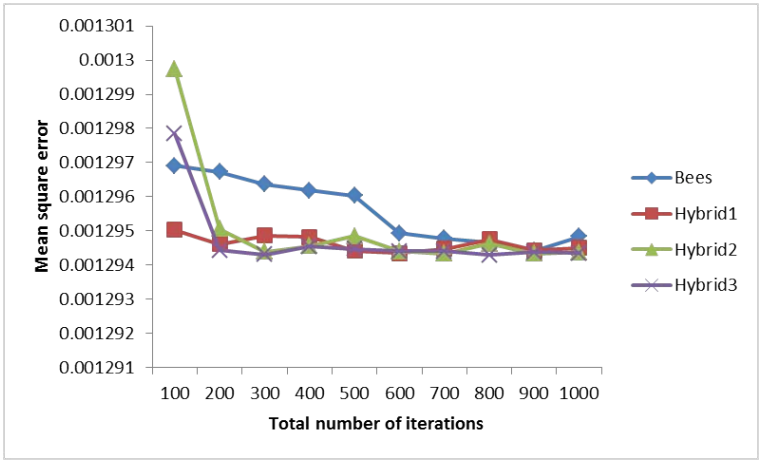


Figure 5.7: Convergence Based on Number of Iterations ($\sigma^2 = 0.00125$)

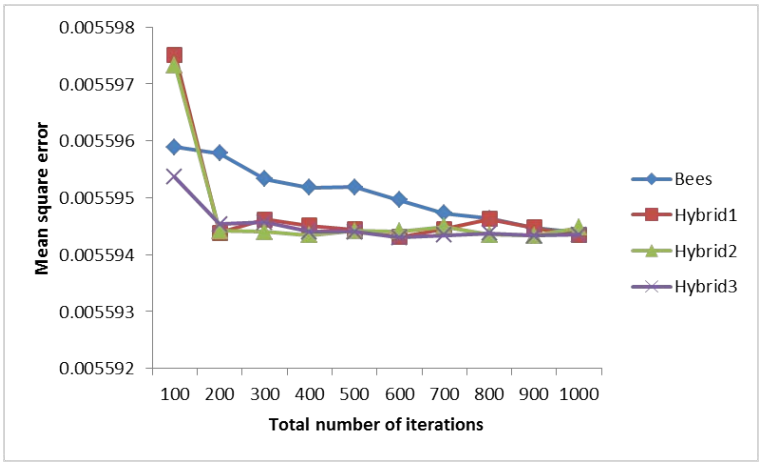


Figure 5.8: Convergence Based on Number of Iterations ($\sigma^2 = 0.00625$)

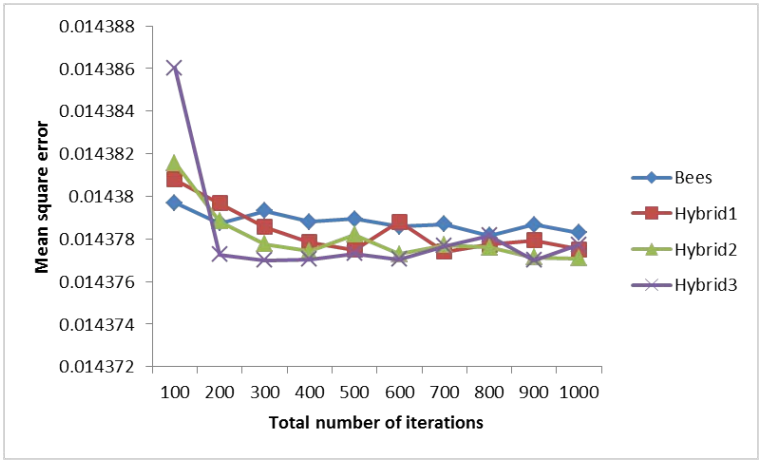


Figure 5.9: Convergence Based on Number of Iterations ($\sigma^2 = 0.01250$)

reducing the iteration for each Bees call and increasing the number of RealPaver calls to keep shrinking the search space does not seem to significantly improve the performance of the search (in comparison with Hybrid1 and Hybrid2). Moreover, using RealPaver to bisect the search space (Hybrid3) improves the performance when the total number of iterations is increased and there is no improvement of using a shrink-Bees process.

Time to Solutions

To get the results in Table 5.5, the execution time of the Bees algorithm is 0.25 seconds for 100 iterations, while the execution time for the genetic algorithm and the gradient descent algorithm are 27 minutes and 1.7 seconds [1], respectively. The mean square errors for the genetic algorithm and the gradient descent algorithm in [1] are 3.5E-3 and 2.6E-3, which are much worse than the results for the genetic algorithm and the gradient descent algorithm in Table 5.5. Table 5.9 shows the CPU benchmarks comparison of two machines we used and [1] used. Although the CPU Mark score is a relative value and real life may not always reflect the CPU Mark, we would say that our machine is about 14.7 times faster than what [1] used. Considering machine performance difference, the Bees algorithm still works lot faster than the genetic algorithm and the gradient descent algorithm. Our experiments also show that when iterations increase from 100 to 1000, the mean square errors of each variance are almost same, although the execution time increases from 0.25 seconds to 2.5 seconds. We can conclude that the Bees algorithm converges quickly and is stable after 100 iterations.

Table 5.9: CPU Benchmarks Comparison [56]

CPU Name	Passmark CPU Mark (higher is better)
Intel Xeon E5540 @ 2.53GHz	4613
Intel Pentium 4 2.40GHz	314

Other Test Examples

Table 5.10: Fuzzy measure to be identified ($n = 5$)

$\mu_1 = 0.1$	$\mu_{12} = 0.3$	$\mu_{123} = 0.5$	$\mu_{1234} = 0.8$
$\mu_2 = 0.2105$	$\mu_{13} = 0.3235$	$\mu_{124} = 0.622$	$\mu_{1235} = 0.9206$
$\mu_3 = 0.2353$	$\mu_{14} = 0.4562$	$\mu_{125} = 0.8667$	$\mu_{1245} = 0.9542$
$\mu_4 = 0.4276$	$\mu_{15} = 0.733$	$\mu_{134} = 0.725$	$\mu_{1345} = 0.9614$
$\mu_5 = 0.6667$	$\mu_{23} = 0.4211$	$\mu_{135} = 0.8824$	$\mu_{2345} = 0.9822$
	$\mu_{24} = 0.48$	$\mu_{145} = 0.8546$	
	$\mu_{25} = 0.8070$	$\mu_{234} = 0.78$	
	$\mu_{34} = 0.5605$	$\mu_{235} = 0.9474$	
	$\mu_{35} = 0.8235$	$\mu_{245} = 0.825$	
	$\mu_{45} = 0.753$	$\mu_{345} = 0.8913$	

Table 5.12 shows the mean square error (E) for problems with 5 criteria using the Bees algorithm and the hybrid approaches with the total of 10000 iterations.

We also tested for the problems with 5 criteria and 6 criteria using the Bees algorithm and adaptive hybrid approaches. Table 5.10 and Table 5.11 show the fuzzy measure values to be identified for problems with 5 or 6 criteria, respectively. These values are generated randomly and follow the constraints we mentioned in Section 3.3.

For each problem, we added 5 different Gaussian noises ($\sigma^2 = 0.0, 0.00096, 0.00125,$ $0.00625, 0.0125$), and for each noise, we generated 3 different size of samples ($m = 120, 180, 240$) and calculated the average as the final result.

The fuzzy measures for $\sigma^2 = 0.0$ for $n = 5$ obtained by using the Bees algorithm are shown in Table 5.13.

When we tested with 6 criteria and evaluated the fitness function, if the monotonicity is

Table 5.11: Fuzzy measure to be identified ($n = 6$)

$\mu_1 = 0.06$	$\mu_{12} = 0.08$	$\mu_{123} = 0.3$	$\mu_{1234} = 0.73$	$\mu_{12345} = 0.99$
$\mu_2 = 0.04$	$\mu_{13} = 0.14$	$\mu_{124} = 0.32$	$\mu_{1235} = 0.5$	$\mu_{12346} = 0.89$
$\mu_3 = 0.08$	$\mu_{14} = 0.15$	$\mu_{125} = 0.34$	$\mu_{1236} = 0.69$	$\mu_{12356} = 0.85$
$\mu_4 = 0.03$	$\mu_{15} = 0.14$	$\mu_{126} = 0.46$	$\mu_{1245} = 0.57$	$\mu_{12456} = 0.97$
$\mu_5 = 0.12$	$\mu_{16} = 0.44$	$\mu_{134} = 0.32$	$\mu_{1246} = 0.88$	$\mu_{13456} = 0.84$
$\mu_6 = 0.03$	$\mu_{23} = 0.22$	$\mu_{135} = 0.26$	$\mu_{1256} = 0.8$	$\mu_{23456} = 0.91$
	$\mu_{24} = 0.26$	$\mu_{136} = 0.49$	$\mu_{1345} = 0.51$	
	$\mu_{25} = 0.14$	$\mu_{145} = 0.34$	$\mu_{1346} = 0.75$	
	$\mu_{26} = 0.33$	$\mu_{146} = 0.67$	$\mu_{1356} = 0.56$	
	$\mu_{34} = 0.14$	$\mu_{156} = 0.5$	$\mu_{1456} = 0.83$	
	$\mu_{35} = 0.19$	$\mu_{234} = 0.72$	$\mu_{2345} = 0.9$	
	$\mu_{36} = 0.22$	$\mu_{235} = 0.41$	$\mu_{2346} = 0.81$	
	$\mu_{45} = 0.26$	$\mu_{236} = 0.58$	$\mu_{2356} = 0.71$	
	$\mu_{46} = 0.17$	$\mu_{245} = 0.41$	$\mu_{2456} = 0.76$	
	$\mu_{56} = 0.41$	$\mu_{246} = 0.37$	$\mu_{3456} = 0.73$	
		$\mu_{256} = 0.48$		
		$\mu_{345} = 0.28$		
		$\mu_{346} = 0.67$		
		$\mu_{356} = 0.43$		
		$\mu_{456} = 0.61$		

Table 5.12: Comparison of Bees algorithm and hybrid approaches for $n = 5$ (Total number of iteration = 10,000)

σ^2	Bees Algorithm	Hybrid1	Hybrid1	Hybrid3
0.0	2.37806E-06	1.8677E-06	2.00346E-06	1.83833E-06
0.00096	0.000762847	0.00076025	0.00076040	0.00076037
0.00125	0.001104373	0.00111413	0.00110764	0.00113659
0.00625	0.004882840	0.00486538	0.00486919	0.00493505
0.01250	0.011163512	0.01159290	0.00925322	0.01176282

Table 5.13: Final obtained fuzzy measure using the Bees algorithm ($n = 5$)

$\mu_1 = 0.0990388$	$\mu_{12} = 0.2975428$	$\mu_{123} = 0.5006999$	$\mu_{1234} = 0.8017974$
$\mu_2 = 0.2086078$	$\mu_{13} = 0.3186199$	$\mu_{124} = 0.6238542$	$\mu_{1235} = 0.9403028$
$\mu_3 = 0.2339973$	$\mu_{14} = 0.4557589$	$\mu_{125} = 0.8650810$	$\mu_{1245} = 0.9549436$
$\mu_4 = 0.4279892$	$\mu_{15} = 0.7374719$	$\mu_{134} = 0.7279080$	$\mu_{1345} = 0.9612262$
$\mu_5 = 0.6668083$	$\mu_{23} = 0.4222468$	$\mu_{135} = 0.8815739$	$\mu_{2345} = 0.9848791$
	$\mu_{24} = 0.4786709$	$\mu_{145} = 0.8510068$	
	$\mu_{25} = 0.8072453$	$\mu_{234} = 0.7816414$	
	$\mu_{34} = 0.5597317$	$\mu_{235} = 0.9392811$	
	$\mu_{35} = 0.8223831$	$\mu_{245} = 0.8253073$	
	$\mu_{45} = 0.7531364$	$\mu_{345} = 0.8929555$	

Table 5.14: Comparison of Bees algorithm and hybrid approaches for $n = 6$ (Total number of iteration = 2000)

σ^2	Bees Algorithm	Hybrid1	Hybrid1	Hybrid3
0.0	1.49199E-07	1.74404E-07	1.75821E-07	1.82291E-07
0.00096	0.00068355	0.00068243	0.00068235	0.00068248
0.00125	0.00070281	0.00070328	0.00070300	0.00070318
0.00625	0.00392318	0.00393209	0.00393518	0.00393419
0.01250	0.00919026	0.00922983	0.00921854	0.00921983

violated, we updated the coefficients of the fuzzy measure by assigning its lower neighbors' maximum value to it rather than add the penalty to the fitness. This makes sure the monotonicity is always followed and increases the chances the fitted fuzzy measures to be found.

Table 5.14 shows the mean square error E for problem with 6 criteria with the total of 2000 iterations. We observe that using hybrid approaches does not guarantee to have better results than using the Bees algorithm. However, using hybrid approaches guarantees the search has focused on relevant parts of the search space and discards irrelevant ones.

The fuzzy measures for $\sigma^2 = 0.0$ for $n = 6$ obtained by using the Bees algorithm are shown in Table 5.15.

Table 5.16 shows the mean square error (E) for problems with 4, 5, and 6 criteria using the Bees algorithm for the total of 1000 iterations. Although the number of variables increases exponentially with the number of criteria (from 14 for $n = 4$ to 62 for $n = 6$), the mean square error E does not become worse, especially for $n = 6$, it's even better.

Table 5.15: Final obtained fuzzy measure using the Bees algorithm ($n = 6$)

$\mu_1 = 0.0602781$	$\mu_{12} = 0.0796912$	$\mu_{123} = 0.2997863$	$\mu_{1234} = 0.7298322$	$\mu_{12345} = 0.9898620$
$\mu_2 = 0.0402176$	$\mu_{13} = 0.1398113$	$\mu_{124} = 0.3201073$	$\mu_{1235} = 0.5006974$	$\mu_{12346} = 0.8900823$
$\mu_3 = 0.0801484$	$\mu_{14} = 0.1493632$	$\mu_{125} = 0.3398449$	$\mu_{1236} = 0.6897268$	$\mu_{12356} = 0.8501863$
$\mu_4 = 0.0298432$	$\mu_{15} = 0.1397904$	$\mu_{126} = 0.4617613$	$\mu_{1245} = 0.5701997$	$\mu_{12456} = 0.9697232$
$\mu_5 = 0.1203329$	$\mu_{16} = 0.4396140$	$\mu_{134} = 0.3197835$	$\mu_{1246} = 0.8799791$	$\mu_{13456} = 0.8401913$
$\mu_6 = 0.0301424$	$\mu_{23} = 0.2198225$	$\mu_{135} = 0.2598456$	$\mu_{1256} = 0.7998698$	$\mu_{23456} = 0.9645371$
	$\mu_{24} = 0.2596472$	$\mu_{136} = 0.4901423$	$\mu_{1345} = 0.5103168$	
	$\mu_{25} = 0.1381568$	$\mu_{145} = 0.3395931$	$\mu_{1346} = 0.7499152$	
	$\mu_{26} = 0.3298391$	$\mu_{146} = 0.6698530$	$\mu_{1356} = 0.5599567$	
	$\mu_{34} = 0.1388819$	$\mu_{156} = 0.4997001$	$\mu_{1456} = 0.8302892$	
	$\mu_{35} = 0.1900780$	$\mu_{234} = 0.7164361$	$\mu_{2345} = 0.9052250$	
	$\mu_{36} = 0.2199434$	$\mu_{235} = 0.4086977$	$\mu_{2346} = 0.8102336$	
	$\mu_{45} = 0.2603797$	$\mu_{236} = 0.5799216$	$\mu_{2356} = 0.7101825$	
	$\mu_{46} = 0.1699762$	$\mu_{245} = 0.4088083$	$\mu_{2456} = 0.7602982$	
	$\mu_{56} = 0.4100879$	$\mu_{246} = 0.3701531$	$\mu_{3456} = 0.7303125$	
		$\mu_{256} = 0.4801795$		
		$\mu_{345} = 0.2784334$		
		$\mu_{346} = 0.6702328$		
		$\mu_{356} = 0.4297651$		
		$\mu_{456} = 0.6087989$		

Table 5.16: Mean Square Error for $n = 4, 5, 6$ by using the Bees algorithm (Total number of iteration = 1000)

σ^2	$n = 4$	$n = 5$	$n = 6$
0.0	2.54E-7	4.02456E-06	8.21607E-07
0.00096	0.0007004	0.000763306	0.000683437
0.00125	0.0012948	0.001106726	0.000707495
0.00625	0.0055944	0.004895794	0.003935055
0.01250	0.0143783	0.011127133	0.009213395

5.2 Interval-Only Technique

5.2.1 Speculative Algorithm

As we described in Chapter 4.1.4, previous attempts of using optimization algorithms for FME have their limitations. Although we originally discarded interval-only optimization techniques for FME due to the size of the problem (size of the search space), here we propose an interval-only technique that primarily focuses on one dimension (the interval-estimated range of the objective function).

As a result, the work presented in this section aims to address the previous-mentioned issues: it guarantees results to be global if results are available, it can factor in interval data, and will not be prone to rounding errors. Our proposed approach is an interval-based technique that we called a speculative algorithm. Instead of using the search space as the primary focus of our search, the proposed algorithm speculates on the value of the objective function before actually arriving to it. The aim of this algorithm is to speed up the search process by primarily focusing on the objective function's range and always betting that the minimum value of the objective function lies in the lower part of the function's range. If our speculation is correct, then we can restrict the search space to the area that allows

the objective function to take on the speculated value, and keep going to the next lower half of the range. In addition, the performance of our algorithm is not compromised: if we speculate wrong, then we do not lose more time than we would have if we had not speculated at all.

In order to verify our speculations on the range of the objective function and guarantee the global results, we use RealPaver. RealPaver [22], as shown in section 5.1.2, is a complete, interval-based, continuous constraint solver able to solve nonlinear systems through interval computations. For the purpose of our speculative algorithm, we primarily use the narrowing function of RealPaver, which discard areas of the search space that are clearly not solution and returns a shrunk search space that contains all solutions.

Eventually, let us call e^* the optimum value of the objective function that is found by our algorithm. e^* is guaranteed to be the global minimum due to the following: our optimistic approach always bets on the lowest values and only "jump" up to larger values if RealPaver has found no feasible element in the search space matching the unrealistically speculated lower value of the objective function, meaning that if RealPaver finds a solution then it is a guaranteed solution, but it can take forever and never find anything.

Let us now take a look at the algorithm in more details. In the case of fuzzy measure extraction, the problem is defined as follows:

- the variables are the coefficients of the fuzzy measure: for n criteria, there are $2^n - 2$ variables, and the initial domain of each variable is $[0, 1]$, which defines the initial search space as $D_0 = [0, 1]^{2^n-2}$;
- the set of constraints is a set of inequality following the monotonicity of fuzzy measure: there are $\sum_{k=1}^{n-2} \binom{n}{k} * (n-2)$ constraints;
- the objective function e is:

$$e = \sum_{j=0}^{m} \left(\tilde{y}_j - \sum_{i=1}^{n} (f(\sigma_j(i)) - f(\sigma_j(i-1)))\mu(A_{(i,j)}) \right)^2 \tag{5.3}$$

The way we deal with e as a constraint sent to RealPaver is by forcing the values of e to be in a specified range R: $e \in R$.

A problem file sent to RealPaver typically consists of the above variables and constraints, in which the initial search space is updated to the current search space and the range of e is the currently speculated one. In the Table 5.17 that follows, we denote such a problem P by $P = D + R$ where D is the current search space and R is the current speculated range of the objective function.

Table 5.17: Our speculative algorithm

// Initial problem $P_0 = D_0 + R_0$

$D_1 = SendToRealPaver(P_0)$ // reduced search space

$[a_1, b_1] = e(D_1)$ //new range of the objective function e

Bisect($[a_1, b_1]$,$R_1 = [a_1, \frac{(a_1+b_1)}{2}]$,$R_2 = [\frac{(a_1+b_1)}{2}, b_1]$)

Create 2 new sub problems: $P_1 = D_1 + R_1$ and $P_2 = D_1 + R_1$

Push P_1 and P_2 into Stack S, with lower-valued P_1 at the top

WHILE S is not empty and no solution is found

    $P = D + R$ problem at the top of S

    $D_1 = SendToRealPaver(P)$ // reduced search space

    IF D_1 is not empty

        $[a_1, b_1] = e(D_1)$ //new range of the objective function e

        IF D_1 is small enough or $[a_1, b_1]$ is small enough

            $[a_1, b_1]$ is our optimum value

        ELSE

            Bisect($[a_1, b_1]$,R_1,R_2)

            Create 2 new sub problems: $P_1 = D_1 + R_1$ and $P_2 = D_1 + R_1$

            Push P_1 and P_2 into S

        ENDIF

    ENDIF

ENDWHILE

5.2.2 Experiments and Results

Quality of Solutions

Table 5.18 shows experimental results for 4 criteria using the speculative algorithm. We compare them with the results of the Bees algorithm, genetic algorithm, and gradient

descent algorithm in the same table. Results of the Bees algorithm, genetic algorithm, and gradient descent algorithm are from 5.1.3. The final obtained fuzzy measures are in Table 5.19.

Table 5.18: Comparison of speculative algorithm and other algorithms ($n = 4$)

σ^2	Gradient Descent [16]	Genetic Algorithm [8]	Bees Algorithm	Speculative Algorithm
0.0	1.4E-7	0.00141472	1.47E-6	3.73E-9
0.00096	0.00083	0.00147230	0.000509	1.55E-5
0.00125	0.01080	0.00141241	0.001106	2.65E-5
0.00625	0.05300	0.00183267	0.004566	0.000705
0.01250	0.10540	0.00241865	0.009878	0.002898

Table 5.19: Final obtained fuzzy measure using speculative algorithm ($n = 4$)

μ	σ^2				
	0.0	0.00096	0.00125	0.00625	0.01250
$\{1\}$	0.1	0.10023	0.10071	0.10001	0.10461
$\{2\}$	0.2105	0.21011	0.20995	0.21253	0.20352
$\{3\}$	0.2353	0.2363	0.23548	0.23436	0.24098
$\{4\}$	0.6667	0.6667	0.66810	0.66488	0.66071
$\{1,2\}$	0.3	0.30006	0.29991	0.30440	0.29790
$\{1,3\}$	0.3235	0.32284	0.32389	0.32506	0.31564
$\{1,4\}$	0.7333	0.73356	0.73178	0.72729	0.74509
$\{2,3\}$	0.4211	0.42075	0.42203	0.42134	0.41666
$\{2,4\}$	0.807	0.80784	0.80511	0.80926	0.80865
$\{3,4\}$	0.8235	0.82299	0.82261	0.81920	0.82530
$\{1,2,3\}$	0.5	0.49991	0.50054	0.49681	0.50624
$\{1,2,4\}$	0.8667	0.86690	0.86715	0.86847	0.86636
$\{1,3,4\}$	0.8824	0.88203	0.88239	0.88885	0.87406
$\{2,3,4\}$	0.9474	0.94774	0.94790	0.94655	0.94404

We observe that when the variance of the Gaussian noise is increased, the mean square error is increased accordingly. However, the mean square error E is always less than the variance, which means our algorithm does not generate negative influence to the data. Furthermore, for all variances, our results are not only much better (closer to 0) than all of the other algorithms but also very close to the optimal results (see Table 5.3).

Other Test Examples

We tested for the problems with 5 criteria and 6 criteria using the speculative algorithm too. Fuzzy measure values to be identified for 5 criteria and 6 criteria are in Table 5.10 and Table 5.11 in Section 5.1.3, respectively. These values are generated randomly and follow the constraints we mentioned in Section 3.3.

For each problem, we also added 5 different Gaussian noises ($\sigma^2 = 0.0, 0.00096, 0.00125, 0.00625, 0.0125$), and for each noise, we generated 3 different size of samples ($m = 120, 180, 240$) and calculated the average as the final result.

Table 5.20 shows the mean square error (E) for problems with 4, 5, and 6 criteria using speculative algorithm. Although the number of variables increases exponentially with the number of criteria (from 14 for $n = 4$ to 62 for $n = 6$), the mean square error E does not become worse, even better in some cases.

Table 5.20: Mean Square Error for $n = 4, 5, 6$ using speculative algorithm

σ^2	$n = 4$	$n = 5$	$n = 6$
0.0	3.73E-9	2.47E-5	3.47E-9
0.00096	1.55E-5	3.30E-5	6.44E-7
0.00125	2.65E-5	3.43E-5	1.13E-6
0.00625	0.0007	0.0004	5.4E-5
0.01250	0.0029	0.00143	0.0026

The fuzzy measures for $\sigma^2 = 0.0$ for $n = 5$ and $n = 6$ obtained by using speculative algorithm are shown in Table 5.21 and Table 5.22, respectively. The obtained fuzzy measure exactly matches the value of actual measures.

Table 5.21: Final obtained fuzzy measure using speculative algorithm ($n = 5$)

$\mu_1 = 0.0992$	$\mu_{12} = 0.2986$	$\mu_{123} = 0.4985$	$\mu_{1234} = 0.8001$
$\mu_2 = 0.2109$	$\mu_{13} = 0.3176$	$\mu_{124} = 0.6219$	$\mu_{1235} = 0.9359$
$\mu_3 = 0.2363$	$\mu_{14} = 0.4562$	$\mu_{125} = 0.8627$	$\mu_{1245} = 0.9558$
$\mu_4 = 0.4272$	$\mu_{15} = 0.7322$	$\mu_{134} = 0.7249$	$\mu_{1345} = 0.9621$
$\mu_5 = 0.6675$	$\mu_{23} = 0.4203$	$\mu_{135} = 0.8790$	$\mu_{2345} = 0.9835$
	$\mu_{24} = 0.4802$	$\mu_{145} = 0.8545$	
	$\mu_{25} = 0.8049$	$\mu_{234} = 0.78$	
	$\mu_{34} = 0.5604$	$\mu_{235} = 0.9359$	
	$\mu_{35} = 0.8235$	$\mu_{245} = 0.825$	
	$\mu_{45} = 0.7528$	$\mu_{345} = 0.8911$	

Table 5.22: Final obtained fuzzy measure using speculative algorithm ($n = 6$)

$\mu_1 = 0.0595$	$\mu_{12} = 0.0805$	$\mu_{123} = 0.2998$	$\mu_{1234} = 0.73$	$\mu_{12345} = 0.99$
$\mu_2 = 0.04$	$\mu_{13} = 0.1401$	$\mu_{124} = 0.32$	$\mu_{1235} = 0.5$	$\mu_{12346} = 0.89$
$\mu_3 = 0.08$	$\mu_{14} = 0.15$	$\mu_{125} = 0.3399$	$\mu_{1236} = 0.69$	$\mu_{12356} = 0.8498$
$\mu_4 = 0.03$	$\mu_{15} = 0.1456$	$\mu_{126} = 0.4602$	$\mu_{1245} = 0.57$	$\mu_{12456} = 0.97$
$\mu_5 = 0.1195$	$\mu_{16} = 0.44$	$\mu_{134} = 0.3201$	$\mu_{1246} = 0.88$	$\mu_{13456} = 0.84$
$\mu_6 = 0.0299$	$\mu_{23} = 0.2201$	$\mu_{135} = 0.2597$	$\mu_{1256} = 0.7948$	$\mu_{23456} = 0.9102$
	$\mu_{24} = 0.26$	$\mu_{136} = 0.4902$	$\mu_{1345} = 0.51$	
	$\mu_{25} = 0.14$	$\mu_{145} = 0.3402$	$\mu_{1346} = 0.75$	
	$\mu_{26} = 0.33$	$\mu_{146} = 0.6703$	$\mu_{1356} = 0.5594$	
	$\mu_{34} = 0.1399$	$\mu_{156} = 0.5012$	$\mu_{1456} = 0.8271$	
	$\mu_{35} = 0.19$	$\mu_{234} = 0.72$	$\mu_{2345} = 0.8995$	
	$\mu_{36} = 0.22$	$\mu_{235} = 0.4098$	$\mu_{2346} = 0.81$	
	$\mu_{45} = 0.2585$	$\mu_{236} = 0.5801$	$\mu_{2356} = 0.71$	
	$\mu_{46} = 0.17$	$\mu_{245} = 0.4107$	$\mu_{2456} = 0.7602$	
	$\mu_{56} = 0.4104$	$\mu_{246} = 0.37$	$\mu_{3456} = 0.7304$	
		$\mu_{256} = 0.4813$		
		$\mu_{345} = 0.2798$		
		$\mu_{346} = 0.67$		
		$\mu_{356} = 0.43$		
		$\mu_{456} = 0.6096$		

Table 5.23 shows the execution time for $n = 4, 5, 6$, and for different number of variables (from 14 to 62), when variance is small, the execution time for $n = 4$ is 10 times faster than the other 2 cases. However, with the increasing of the variance, the execution times for different number of variables are very close.

Table 5.23: Average execution time for speculative algorithm (s)

σ^2	$n = 4$	$n = 5$	$n = 6$
0.0	14.6	138.7	114.7
0.00096	27.4	137.0	140.3
0.00125	32.2	166.1	281.5
0.00625	240.0	383.1	473.3
0.01250	606.9	962.7	334.1

Summary. In this chapter, we described three algorithms: our version of the Bees algorithm, an adaptive hybrid solving approach, and our speculative algorithm to fuzzy measure extraction (FME). We also described and reported the results on our experiments: they indicate that the proposed algorithms yield promising performance, with similar or better accuracy than results obtained using algorithms previously applied to FME and described in Section 4.1. Using RealPaver, a complete interval solver, we were able to guarantee that the search focused on relevant parts of the search space and discarded irrelevant ones. In the next chapter, we will show how our contributed algorithms were applied to real-world situations.

Chapter 6

Contributed Applications and Results

In chapter 5, we showed that our algorithms yielded promising performance in the case of toy examples. In this chapter, we apply these algorithms to real applications.

In Section 6.1, we describe the software quality assessment problem: we show how we model it as a MCDM problem, and then show how the Bees algorithm and the adaptive hybrid approach can be used to solve this problem. In Section 6.2, we consider at-risk students prediction as a MCDM problem and show how the Bees algorithm apply to it.

To evaluate the accuracy of a prediction of software quality or student risk level, we compared the results of the prediction with the actually observed values (sample data).

In both cases, the observed values y may be different from even with the exact same values of inputs $x_1, \ldots, x_n$, because in reality, the values y depend not only on the values of $x_1, \ldots, x_n$, but also on many other values which we do not measure or take into account. It is often reasonable to assume that for each combination of x_i values, possible values of y are normally distributed, with some mean μ and standard deviation σ. In many practical situations, we only predicted a single value $\widetilde{y}$. To gauge the accuracy of this prediction based on the observations, we used crisp approach and fuzzy approach. In crisp evaluation, a method is accurate if $\widetilde{y}$ belongs to a k_0-sigma interval $[\mu - k_0 \cdot \sigma, \mu + k_0 \cdot \sigma]$, for some pre-selected value k_0 (e.g., 2, 3, or 6). In fuzzy approach, we estimate the degree to which, for given μ and σ, an estimate $\widetilde{y}$ is a "typical" representative of the corresponding Gaussian random variable.

6.1 Software Quality Assessment

Software quality is defined as "Conformance to explicitly stated functional and performance requirements, explicitly documented development standards and implicit characteristics that are expected of all professionally developed software" [52]. This definition addresses two aspects of software quality.

The first is conformance to explicitly stated functional and performance requirements and explicitly documented development standards. These can be found in the requirements document developed between the customer and client. For example, if the software product is a database, the requirements might include an item such as "the table must have seven columns containing data described by the type integer". This requirement can be measured by counting the number of columns and comparing the data type to that stated in the requirements data. The performance requirements are also measurable. For example, the time a software product takes to complete a given task is a measure of performance. Functional and performance requirements and explicitly documented development standards are thus measurable.

The second aspect of software quality addressed in this definition is the implicit characteristics of all professionally developed software. These are characteristics such as reusability or flexibility. These implicit characteristics are also known as quality factors. Most software engineers, programmers and managers believe that software quality factors are best judged by experts. However, research has shown that expert judgments are often inconsistent and subjective. For example, experts such as Lorenz and Kidd considered multiple inheritances (software property) as a sign of bad quality code, but multiple inheritance is widely accepted in the programming community [35]. Inconsistent expert opinion makes judgment of implicit characteristics such as reusability difficult. Process improvement strategies such as the Capability Maturity Model Integration (CMMI) have shown to reduce software quality costs [49]. CMMI provides software management with five levels of process capability and maturity. To achieve each level of maturity, management needs

to fulfill a set of goals. In fact, CMMI level 4 requires software measurement to be predictive of quality. CMMI level 4 is easily applied for the explicitly stated functional and performance requirements, but it is difficult to apply for implicit characteristics or quality factors. There is no quantitative measurement for quality factors such as reusability. This subjective aspect of software quality results in making software quality management difficult.

Although this process is difficult, many companies depend upon software to acquire a competitive edge. CIOs and IT leaders often assume the roles of change and risk managers in Fortune 500 companies [73]. CIOs and IT leaders are expected to facilitate business problem solving as well as provide technical solutions. These leaders are required to meet market expectations and performance goals through the use of the latest software technologies. The estimation of cost, schedule and quality goals is particularly difficult with relation to software projects. In the widely recognized software engineering textbook [52], the author says "A software project manager is confronted with a dilemma at the very beginning of a software engineering project. Quantitative estimates and an organized plan are required, but solid information is unavailable" [52].

Solid information regarding the quality of the software product is difficult to estimate. There is currently no tool or process that uses quantitative information to calculate quality factors for software products. This lack of solid information creates problems with software project management. Sommerville gives three reasons why software project management activities are difficult when compared to other engineering project management activities [57]. The three reasons are

1. "*The product is intangible*" [57]: Other than counting lines of code, there is no way to track the progress of the project because software product cannot be seen or touched.

2. "*There are no standard software processes*" [57]: Since software engineering is a relatively new engineering practice, there is no accepted industry-wide standard software process.

3. "*Large software projects are usually different in some way from the previous projects*"- [57]: Software technologies are changing continuously because of this continuous change of technology, and experience transfer from one project to another project is non-existent.

To understand the subjective estimates and the measurements, it is important to understand software quality research.

Theoretical models define several quality factors such as reusability and flexibility but do not quantify them. Predictive models are models based on statistical techniques that predict characteristics such as fault density or fault proneness using direct measurements from code (product metrics). Predictive models predict the faults but have no theoretical evidence to support causality. One solution to remedy this problem is to add prediction capability to a theoretical model. Quality factors defined in a theoretical model are not measurable and hence cannot be predicted. One theoretical software quality model that defined the quality factors and linked all of them to measurable metrics in object-oriented software was Bansiya and Davis's model [3]. Bansiya and Davis' model is more complete than other theoretical software quality models, so it was chosen for analysis here. Bansiya and Davis's model defines the quality factors and links them to QMOOD set of metrics defined in Table 6.1 and Table 6.2.

Although the Bansiya and Davis model provides a solid explanation for the design quality of object-oriented design, it presents some limitations. The major problems with the Bansiya and Davis model are their validation process, data used in the research and lack of prediction capability [43].

According to Moody [43], the major problem with the Bansiya and Davis model is the validation process. Validation of the model was performed using a case study and a lab experimental approach which involves gathering expert opinion of software quality. Elicitation of expert opinion is a very well defined process that includes calculation of reliability and inter-rater agreement [5, 32]. Bansiya and Davis [3] appear to have ignored the guidelines on elicitation of expert opinion in the validation process of the model. They

Table 6.1: QMOOD model [3]

Quality Factor Definition	Bansiya and Davis's Model
Reusability Reflects the presence of object oriented design characteristics that allow a design to be reapplied to a new problem without significant effort.	-0.25 * Coupling + 0.25 * Cohesion + 0.5*Messaging +0.5* Design Size
Flexibility Characteristics that allow the incorporation of changes in a design. The ability of a design to be adapted to provide functionality related capabilities.	0.25 * Encapsulation - 0.25 * Coupling + 0.5 * Composition + 0.5 * Polymorphism
Understandability The properties of designs that enable it to be easily learned and comprehended. This directly relates to the complexity of design structure.	-0.33 * Abstraction + 0.33 * Encapsulation -0.33 * Coupling + 0.33 * Cohesion -0.33 * Polymorphism - 0.33 * Complexity - 0.33 * Design Size
Functionality The responsibility assigned to the classes of a design, which are made available by classes through their public interfaces.	0.12 * Cohesion + 0.22 * Polymorphism + 0.22 * Messaging + 0.22 * Design Size + 0.22 * Hierarchies
Extendibility Refers to their presence and usage of properties in an existing design that allow for the incorporation of new requirements in the design.	0.5 * Abstraction - 0.5 * Coupling + 0.5 * Inheritance +0.5 * Polymorphism
Effectiveness This refers to the designs ability to achieve the desired functionality and behavior using object oriented design concepts and techniques.	0.2 * Abstraction + 0.2 * Encapsulation + 0.2*Composition + 0.2 * Inheritance + 0.2 * Polymorphism

Table 6.2: QMOOD Metric Definitions

Design Size (DSC)
A measure of number of classes used in the design.
Hierarchies (NOH)
Hierarchies are used to represent different generalization-specialization aspects of the design.
Abstraction (ANA)
A measure of generalization-specialization aspect of design.
Encapsulation (DAM)
Defined as the enclosing of data and behavior within a single construct.
Coupling (DCC)
Defines the inter dependency of an object on other objects in a design.
Cohesion (CAM)
Accesses the relatedness of methods and attributes in a class.
Composition (MOA)
Measures the "part-of", "has", "consists-of", or "part-whole" relationships, which are aggregation relationships in object oriented design.
Inheritance (MFA)
A measure of the "is-a" relationship between classes.
Polymorphism (NOP)
It is a measure of services that are dynamically determined at run-time in an object.
Messaging (CIS)
A count of number of public methods those are available as services to other classes.
Complexity (NOM)
A measure of the degree of difficulty in understanding and comprehening the internal and external structure of classes and their relationships.

did not develop a common set of heuristics or a questionnaire to collect the expert opinion. There was no calculation of agreement among experts on their opinion of a package or a quality factor.

In addition, a major issue with the model is that the packages used in the study were not very different from each other. The COOL packages used by Bansiya and Davis [3] were developed by different teams, but had the same set of requirements. The methodology developed to calculate quality factors based on the QMOOD set of metrics was restricted to the COOL packages. This allowed the researchers to normalize the data but restricted the prediction capability of the model. Thus, the model could not be used to calculate or compare quality factors for software packages with different sets of requirements. Kitchenham et al. [32] provide a framework of software measurement validation. Their opinion on prediction models in software is that they are based only on empirical evidence and not on theoretical evidence. This restricts the models from developing a causal relationship to quality factors. Because the Bansiya and Davis model was developed as a theoretical model, it implies the causal relationship between design characteristics and quality factors. The validation of the model with data, however, lacked rigor. Etzkorn et al. [14] developed a model to predict reusability of object-oriented code based on theoretical evidence, software metrics and expert opinion. Etzkorn et al. established a relationship between software metrics and reusability; they thus provide a basic methodology to use expert opinion in evaluation of software quality factors. Virani et al. [63] added prediction capability to the Bansiya and Davis model using a standard machine learning algorithm C4.5.

Machine learning has established itself as an important technique for predicting software product quality because the more a classifier can learn, the better decisions it will make in building a predictive model [39]. Osbeck et al. improved the prediction capability using J48, Part, and Random Forest, and the ensemble learning techniques examined were boosting, bagging, and stacking [47].

6.1.1 Experiments and Results on SQA

Testing Methodology

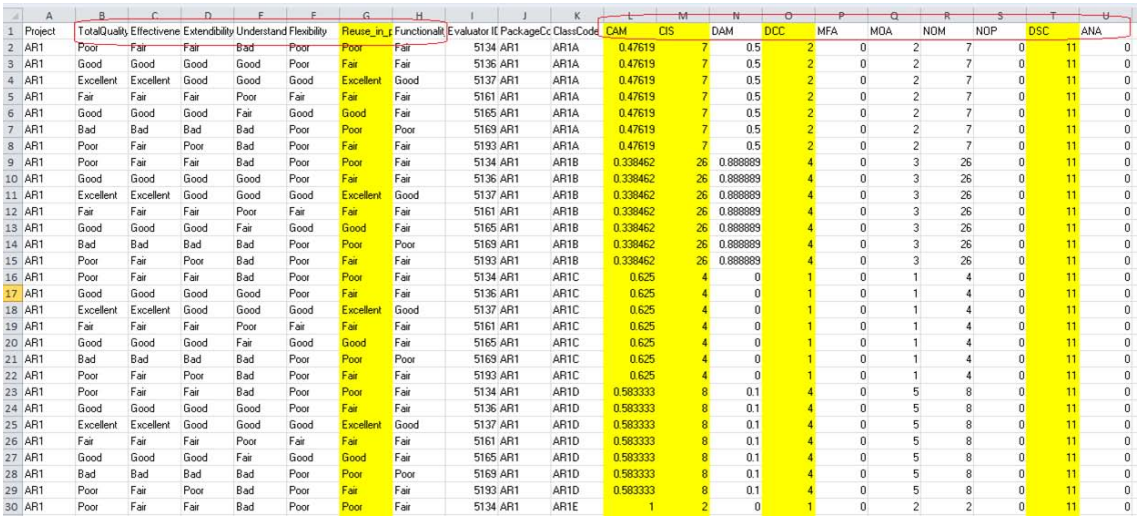
In the previous chapter, we have shown that hybrid approaches help to improve the quality of the extracted fuzzy measure. We now aim at assessing the performance of our approach in automatically predicting software quality. The metrics we use are the Quality Model for Object-Oriented Design (QMOOD) metrics, as defined in [3] and used by Virani et al [63]. We focus on the metrics and the quality factors related to QMOOD model, as in [3], as defined in Table 6.1 and Table 6.2.

The data at our disposal to run our tests comes from 31 software packages and is composed of 2330 samples. The example of the SQA sample data is in Figure 6.1. Each package was rated individually by a group of experts for each of the metrics criteria and for the total quality. The rate for each quality factor is qualitative; in the SQA case, the options are bad, poor, fair, good, and excellent. The rate for each metrics is quantitative, either real number or integer, range from 0 to a couple of hundreds. Note that the data is the same data set as used in [47], in which the authors used WEKA, a machine learning tool to model and predict SQA with the same rating classes.

Using the above-described data, our goal is to show that: (1) our approach allows to accurately recreate decisions (data) used to determine the reasoning process (fuzzy measure); and that (2) it also works to predict decisions (data) that were not included in determining the reasoning process (fuzzy measure) but that were available to us.

Experimental Results

Since the rate for each sample is quantitative, and the metrics may have the different unit, we need to transform them into a common scale. For qualitative quality rating, we set 0 for bad, 0.25 for poor, 0.5 for fair, 0.75 for good, and 1 for excellent. For each metrics, we use the utility function to transform initial evaluation values into the common scale, that is, the real numbers between 0 and 1.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Project	TotalQuality	Effectivene	Extendibility	Understand	Flexibility	Reuse_in_c	Functionalit	Evaluator IE	PackageCo	ClassCode	CAM	CIS	DAM	DCC	MFA	MOA	NOM	NOP	DSC	ANA
2	AR1	Poor	Fair	Fair	Bad	Poor	Poor	Fair	5134	AR1	AR1A	0.47619	7	0.5	2	0	2	7	0	11	0
3	AR1	Good	Good	Good	Good	Poor	Fair	Fair	5136	AR1	AR1A	0.47619	7	0.5	2	0	2	7	0	11	0
4	AR1	Excellent	Excellent	Good	Good	Good	Excellent	Good	5137	AR1	AR1A	0.47619	7	0.5	2	0	2	7	0	11	0
5	AR1	Fair	Fair	Fair	Poor	Fair	Fair	Fair	5161	AR1	AR1A	0.47619	7	0.5	2	0	2	7	0	11	0
6	AR1	Good	Good	Good	Fair	Good	Good	Fair	5165	AR1	AR1A	0.47619	7	0.5	2	0	2	7	0	11	0
7	AR1	Bad	Bad	Bad	Bad	Poor	Poor	Poor	5169	AR1	AR1A	0.47619	7	0.5	2	0	2	7	0	11	0
8	AR1	Poor	Fair	Poor	Bad	Poor	Fair	Fair	5193	AR1	AR1A	0.47619	7	0.5	2	0	2	7	0	11	0
9	AR1	Poor	Fair	Fair	Bad	Poor	Poor	Fair	5134	AR1	AR1B	0.338462	26	0.888889	4	0	3	26	0	11	0
10	AR1	Good	Good	Good	Good	Poor	Fair	Fair	5136	AR1	AR1B	0.338462	26	0.888889	4	0	3	26	0	11	0
11	AR1	Excellent	Excellent	Good	Good	Good	Excellent	Good	5137	AR1	AR1B	0.338462	26	0.888889	4	0	3	26	0	11	0
12	AR1	Fair	Fair	Fair	Poor	Fair	Fair	Fair	5161	AR1	AR1B	0.338462	26	0.888889	4	0	3	26	0	11	0
13	AR1	Good	Good	Good	Fair	Good	Good	Fair	5165	AR1	AR1B	0.338462	26	0.888889	4	0	3	26	0	11	0
14	AR1	Bad	Bad	Bad	Bad	Poor	Poor	Poor	5169	AR1	AR1B	0.338462	26	0.888889	4	0	3	26	0	11	0
15	AR1	Poor	Fair	Poor	Bad	Poor	Fair	Fair	5193	AR1	AR1B	0.338462	26	0.888889	4	0	3	26	0	11	0
16	AR1	Poor	Fair	Fair	Bad	Poor	Poor	Fair	5134	AR1	AR1C	0.625	4	0	1	0	1	4	0	11	0
17	AR1	Good	Good	Good	Good	Poor	Fair	Fair	5136	AR1	AR1C	0.625	4	0	1	0	1	4	0	11	0
18	AR1	Excellent	Excellent	Good	Good	Good	Excellent	Good	5137	AR1	AR1C	0.625	4	0	1	0	1	4	0	11	0
19	AR1	Fair	Fair	Fair	Poor	Fair	Fair	Fair	5161	AR1	AR1C	0.625	4	0	1	0	1	4	0	11	0
20	AR1	Good	Good	Good	Fair	Good	Good	Fair	5165	AR1	AR1C	0.625	4	0	1	0	1	4	0	11	0
21	AR1	Bad	Bad	Bad	Bad	Poor	Poor	Poor	5169	AR1	AR1C	0.625	4	0	1	0	1	4	0	11	0
22	AR1	Poor	Fair	Poor	Bad	Poor	Fair	Fair	5193	AR1	AR1C	0.625	4	0	1	0	1	4	0	11	0
23	AR1	Poor	Fair	Fair	Bad	Poor	Poor	Fair	5134	AR1	AR1D	0.583333	8	0.1	4	0	5	8	0	11	0
24	AR1	Good	Good	Good	Good	Poor	Fair	Fair	5136	AR1	AR1D	0.583333	8	0.1	4	0	5	8	0	11	0
25	AR1	Excellent	Excellent	Good	Good	Good	Excellent	Good	5137	AR1	AR1D	0.583333	8	0.1	4	0	5	8	0	11	0
26	AR1	Fair	Fair	Fair	Poor	Fair	Fair	Fair	5161	AR1	AR1D	0.583333	8	0.1	4	0	5	8	0	11	0
27	AR1	Good	Good	Good	Fair	Good	Good	Fair	5165	AR1	AR1D	0.583333	8	0.1	4	0	5	8	0	11	0
28	AR1	Bad	Bad	Bad	Bad	Poor	Poor	Poor	5169	AR1	AR1D	0.583333	8	0.1	4	0	5	8	0	11	0
29	AR1	Poor	Fair	Poor	Bad	Poor	Fair	Fair	5193	AR1	AR1D	0.583333	8	0.1	4	0	5	8	0	11	0
30	AR1	Poor	Fair	Fair	Bad	Poor	Poor	Fair	5134	AR1	AR1E	1	2	0	1	0	2	2	0	11	0

Figure 6.1: SQA Sample Data

We then applied the same 3 configurations on SQA data: Hybrid1, Hybrid2, Hybrid3, together with the Bees algorithm. Each configuration was run 10 times for each property factor, and we calculated the average as the final results. To be able to compare against the performance of the Bees algorithm, each Bees algorithm in Hybrid1, Hybrid2 and Hybrid3 was run 1000 times, and the overall number of iterations in Hybrid2 and Hybrid3 was 10000.

Table 6.3: Comparison with the other algorithms

Quality Factor	Bees Algorithm	Hybrid1	Hybrid2	Hybrid3
Reusability	0.076735	0.076753	0.076736	**0.076713**
Flexibility	0.098318	0.099891	0.097946	**0.097942**
Extendibility	0.104124	0.104836	0.104142	**0.104117**
Functionality	0.072958	0.073175	0.072948	**0.072918**

Values of e, as defined in Section 3, are reported in Table 6.3. We can observe that

Hybrid3 is slightly more efficient than Hybrid1 and Hybrid2. Overall, Hybrid3 is as efficient as the Bees algorithm alone, which indicates that, in general, the Bees algorithm does not fall into local minima for the tests we ran: the advantage of Hybrid3 over Bees is that Hybrid3 will guarantee that the search is indeed global, as opposed to the Bees, which will not come with any guarantee.

In addition, let us note that, within 1000 iterations, the Bees algorithm was not able to find the optimal fuzzy measures for Effectiveness and Understandability that satisfied all constraints.

Discussion of quality assessment

We used the obtained fuzzy measures to recreate the experts' decision (our seed data). We compared the evaluations we obtained to the original experts' decisions and assessed the accuracy of our approach using three different evaluation processes.

As we mentioned earlier, the sample data set consists of 31 software packages and 2330 samples. For reusability, these 2330 samples can be grouped into 264 groups by input values (DCC, CAM, CIS and DSC, as we described in Table 6.2). In each group, with the same input values, experts' decisions are distributed among five rating classes (bad, poor, fair, good, excellent). For example, for software package AR1 with the following metrics values: DCC=0, CAM=0, CIS=0, and DSC=11, 14 experts evaluated it. Four experts ranked it as Poor, six experts ranked it as Fair, two experts ranked it as Good, two experts ranked it as Excellent and no expert ranked it as Bad, as illustrated in Figure 6.2. Since the decisions from different experts are not consistent, it is debatable what an accurate prediction would be. We explored three ways to assess accuracy, which we describe later. For instance, we computed the average assessment μ and the standard deviation σ over the known expert's decisions for each group, as shown in Figure 6.3. In the previous example, the average rank of 14 experts is 0.535714 and the standard deviation is 0.256776. If the calculated Choquet integral value is close to 0.535714, then we can say the predicated value is accurate. But if calculated Choquet integral value is 0.2, is this predicated value accurate? To evaluate

how accurate the predicated value is, we use the following approaches.

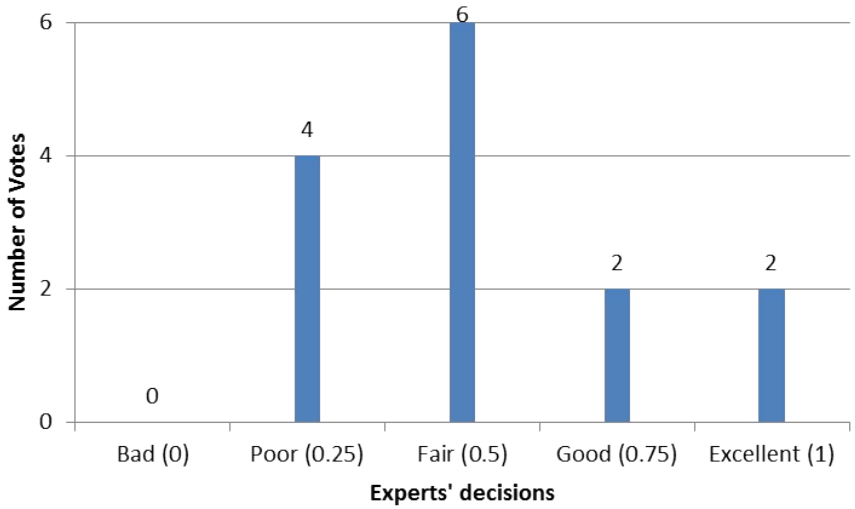


Figure 6.2: SQA Sample Data – Experts' Decisions

						0	0.25	0.5	0.75	1	Original	
Project	DCC	CAM	CIS	DSC	#samples	Bad	Poor	Fair	Good	Excellent	Avg	stdev
AR1	0	0	0	11	14	0	4	6	2	2	0.535714	0.256776
AR1	0	0.5	18	11	7	0	2	3	1	1	0.535714	0.267261
AR1	1	0.464286	7	11	7	0	2	3	1	1	0.535714	0.267261
AR1	1	0.625	4	11	7	0	2	3	1	1	0.535714	0.267261
AR1	1	0.6875	8	11	7	0	2	3	1	1	0.535714	0.267261
AR1	1	1	2	11	7	0	2	3	1	1	0.535714	0.267261
AR1	2	0.47619	7	11	7	0	2	3	1	1	0.535714	0.267261
AR1	3	0.666667	3	11	7	0	2	3	1	1	0.535714	0.267261
AR1	4	0.338462	26	11	7	0	2	3	1	1	0.535714	0.267261
AR1	4	0.583333	8	11	7	0	2	3	1	1	0.535714	0.267261

Figure 6.3: SQA Sample Data – Group by Input Values

1. Eval1: For each group, we calculated how close the predicted decisions (calculated using Choquet integral) are to the known experts' decisions using Gaussian distribution $f = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$. We know that if the predicated value x is close to μ then $f(x)$ is close to the peak $f(\mu)$, so we use $f(x)/f(\mu)$ to calculate the percentage of accuracy for each predicted decision. The closer the predicted decision is to the average assessment, the higher the accuracy is, as shown in Figure 6.4 and Figure 6.5. For

the example we have shown above, $f(\mu) = 1.553657$, the calculated Choquet integral $x = 0.117269$ and $f(x) = 0.411806$, then the percentage of accuracy is 26.5%.

						0	0.25	0.5	0.75	1	Original			Calculate f		Eval 1
Project	DCC	CAM	CIS	DSC	#samples	Bad	Poor	Fair	Good	Excellent	Avg	stdev	Hybrid3	Original	Hybrid3	Accuracy
AR1	0	0	0	11	14	0	4	6	2	2	0.535714	0.256776	0.117269	1.553657	0.411806	0.265056
AR1	0	0.5	18	11	7	0	2	3	1	1	0.535714	0.267261	0.281	1.492705	0.947845	0.634985
AR1	1	0.464286	7	11	7	0	2	3	1	1	0.535714	0.267261	0.275167	1.492705	0.92811	0.621764
AR1	1	0.625	4	11	7	0	2	3	1	1	0.535714	0.267261	0.355347	1.492705	1.188704	0.796342
AR1	1	0.6875	8	11	7	0	2	3	1	1	0.535714	0.267261	0.386528	1.492705	1.27736	0.855735
AR1	1	1	2	11	7	0	2	3	1	1	0.535714	0.267261	0.542435	1.492705	1.492233	0.999684
AR1	2	0.47619	7	11	7	0	2	3	1	1	0.535714	0.267261	0.30048	1.492705	1.013334	0.678857
AR1	3	0.666667	3	11	7	0	2	3	1	1	0.535714	0.267261	0.414884	1.492705	1.347689	0.90285
AR1	4	0.338462	26	11	7	0	2	3	1	1	0.535714	0.267261	0.270518	1.492705	0.912367	0.611217
AR1	4	0.583333	8	11	7	0	2	3	1	1	0.535714	0.267261	0.392684	1.492705	1.293545	0.866578

experts' decisions

predicted decisions

Figure 6.4: SQA Assessment – Eval1 (data)

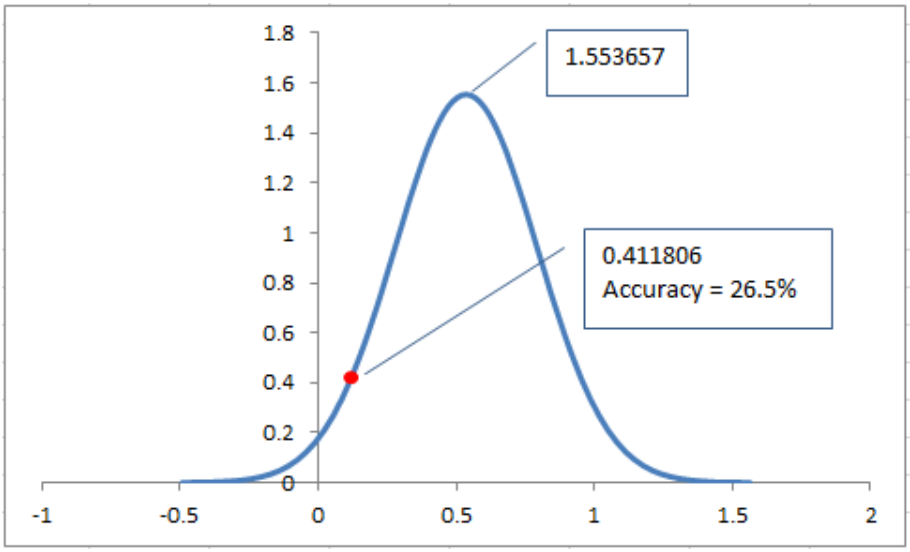


Figure 6.5: SQA Assessment – Eval1 (plot)

2. Eval2: We used the same average as for Eval1 but allowed for some flexibility by accepting any evaluation within σ (standard deviation) of the target average. This evaluation accounted for the uncertainty of the experts' decisions. Every predicted decision between $\mu - \sigma$ and $\mu + \sigma$, that is, $x \in [\mu - \sigma, \mu + \sigma]$, is considered to be an accurate decision. In the previous example, since $x = 0.117269 \notin [0.278938, 0.792490]$, then it is not an accurate decision.

3. Eval3: Instead of performing the assessment based on the average, we assume that the maximum number of votes for one rating would correspond to the 100% accuracy, and the accuracy for the other possible ratings depends on the number of votes for them (as compared to the 100% accuracy). For instance, for AR1 example as described before, the max number of votes was for Fair. Hence a 100% accuracy is given to Fair evaluation. We then mapped the number of experts' decisions from 0% to 100% based on the maximum number of votes for one rating (which would be the one to receive 100% accuracy) and interpolated all other possible ratings (numerical values in between posted ratings) to determine their accuracy. As a result, since Bad did not get any vote, an evaluation to Bad would get a 0% accuracy. Since Poor received 4 votes, any evaluation to Poor would be $4/6 = 66.7\%$ accurate. Any evaluation falling in between would be rated based on the interpolation to closest rating. The closer the predicted decision is to the majority rating, the higher the accuracy is, as shown in Figure 6.6.

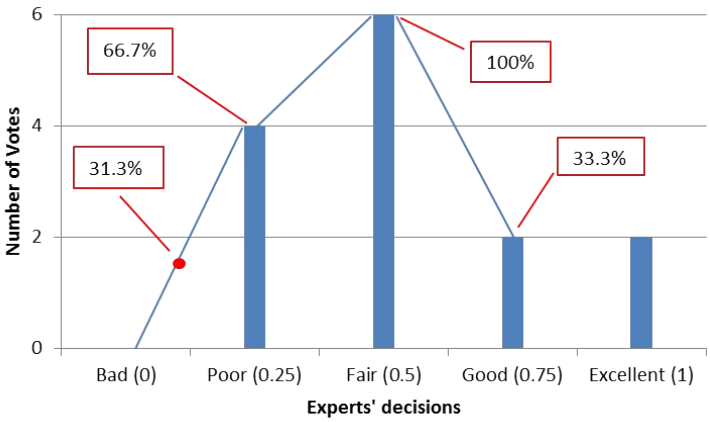


Figure 6.6: SQA Assessment – Eval3

The overall evaluation results for all quality factors using Hybrid3 are in Table 6.4; we also list and compare results obtained using the machine learning approach used in [47]. In [47], a ten-fold cross validation technique is used to train and test the model. The

accuracy is calculated by dividing the total number of instances classified correctly by the total number of samples [47].

Table 6.4: Accuracy using Hybrid3

Quality Factor	Traditional Machine learning approach [47]	Eval 1 (Average)	Eval 2 ($\mu \pm \sigma$)	Eval 3
Reusability	62.08%	76.91%	79.17%	67.80%
Flexibility	62.12%	73.52%	74.33%	65.97%
Extendibility	63.84%	75.26%	76.65%	69.88%
Functionality	66.14%	61.71%	57.68%	58.99%

The accuracy we reached using either Eval1 or Eval2 were similar. Considering the same piece of software, experts often disagree on the decisions to make, sometimes radically. As a result, the reported accuracy of our approach using this evaluation may be affected. We observe, in particular, that when allowing some uncertainty (also seen as flexibility), the reported accuracy of our approach improved.

We then considered the distribution of all decisions and evaluated whether the measured quality matched the majority decisions (Eval3). Although the accuracy reported when using Eval3 does not match that of Eval1 and Eval2, it still comes close to that obtained through machine learning and we believe that Eval3 is more relevant to group decision than the other two.

6.2 At-Risk Students Prediction

Today, more and more students attend colleges. Between fall 2000 to fall 2010, the total undergraduate enrollment in degree-seeking universities increased by 37 percent, from 13.2 million to 18.1 million. The number of degrees earned in the same time period increased

by 33 percent for bachelor's degrees, from 1.24 million to 1.65 million [2]. The 6-year graduation rate for first-time, full-time, degree-seeking students who began academic study at a 4-year institution in fall 2004 is 58 percent. In comparison, the 6-year graduation rate for first-time, full-time, degree-seeking students starting in fall 1996 is 55 percent [2]. Currently, improving students' retention and graduation rate is an important goal for many institutions of higher education.

One way institutions can work to improve the retention and graduation rates is to analyze students' performance during their first-year in college and classify their departure risk level. This allows the institution to take necessary actions to address students' needs during their early academic period to reduce the chance of departure and therefore to improve their retention and graduation rate.

In this study, we rely on existing research for classifying at-risk students based on students' academic performance. Experts are used to evaluating students' academic performance and then to classifying the at-risk students, but it is desired that this process could be automated so as to ensure the consistency of the classification as well as its timeliness. In this article, we show that the performance prediction can be viewed as a multi-criteria decision-making (MCDM) problem, and we propose to extract experts' decision process using Fuzzy Measure Extraction for MCDM.

Students' departure from college can be defined as students' stop out (students withdrew from the university for one academic year or less [59]), drop out (students withdrew from the university for more than one academic year [59]), or transfer to other institutions, which causes a significant loss for both the student and the university [74]. Several reasons may cause students' departure from the college, such as, not being prepared for the academic study, poor academic performance, lack of proper study attitudes or motivation, inability to cope with the competing demands of study, lack of financial support, working too long at jobs, and so on [6, 28]. Academic performance, such as GPA, is a significant predictor for retention [28]. Students with a lower GPA are statistically more likely to leave college than those with a higher GPA.

According to past research, several major factors may affect the academic performance, as described in [74, 34]:

- **Prior academic achievement.** This includes high school performance, state test performance, major area (math, reading, writing) preparation, and so on. High-school performance has been shown to be significantly related to college persistence [34]. It is a strong predictor of first-year college performance. Students who complete high-level mathematics class in high school have more chance to retain and graduate from college than those who do not complete those courses [34]. Students' math placement score that took prior to matriculation also related to persistence [48]. Students with a higher math placement score take less remedial courses than those who with lower math placement score, therefore increase the possibility of retention and graduation.

- **Not entering college directly after high school.** This is also called delayed matriculation. Students with delayed matriculation usually enter college after a period of non-enrollment following their high-school graduation [74]. They have a much higher rate of departure from college comparing to students who matriculated immediately after high school graduation [58].

- **Family income.** Students with higher family income are more likely to attend the college or university, showing better preparation for higher level study. On the other hand, students from low-income backgrounds have higher college-departure rates than students from high-income backgrounds [74, 34].

- **Employment intentions.** Research has shown that student employment may hinder academic success by reducing the amount of time students have to devote to their studies and impede students' integration into the campus community [74].

- **Being a first-generation college student.** Parents' education level has been shown to be an important variable to model college student departure [58, 74]. First-

generation students had lower educational aspirations than continuing-generation students [34].

The goal of this study is to classify students' risk levels in their first year's performance; therefore, we try to learn from existing students data to find out the potential relation between students' academic performance and the factors that affect the persistence (predictor variables).

In [74], a Cox proportional hazards (PH) regression model is used to derive a risk score for each student and to classify students into three distinct risk categories: low, medium, and high risk.

As hinted earlier, students risk level predictions can be seen as a Multi-Criteria Decision-Making problem. The straightforward reason for that is the following: the student's risk level can be identified according to his/her academic performance (i.e., cumulative GPA (decision) of a student), which is related to a set of predictor variables (i.e., multiple criteria that affect student's performance). Existing students data can be used to find the patterns (fuzzy measures) between the academic performance (GPA) and a set of predictor variables. As a result, these patterns can be used to predict the future students' performance, and then classify the risk level of the student.

As a result, based on what we presented about fuzzy measures and Choquet integrals, if we can find an appropriate fuzzy measure μ, predicting the student performance can be expressed as follows:

$$\boxed{\text{Performance(student } A) = \text{Choquet}(\mu, \text{PV}(A))}$$

note: PV is the abbreviation of predictor variables.

Based on the above formula, we focus on determining μ, which can be viewed as a quantitative model for the expert's decision-making process. It is obtained from seed data, namely sample data of experts' decisions with respect to known pieces of student information. As a result, in the above 3-body problem, two of the components are known: the expert's decision and the set of predictor variables, and we aim at determining the

matching μ.

In practice, we have access to a number of such expert(s)' decision values along with the corresponding sets of metrics values for different students A_i. As pointed out earlier, in Section 3, determining μ consists of solving the following problem:

$$\begin{boxed}\min e = \sum_{A_i} \left(\text{performance}(A_i) - \text{Choquet}(\mu, \text{PV}(A_i))\right)^2 \\ \text{s.t. } \mu \text{ satisfies monotonicity constraints}\end{boxed}$$

6.2.1 Experiments and Results

What do we want to show and how?

Through our experiments, we aim at quantifying the performance of our approach in automatically predict students risk level through their first-year academic performance. As hinted before, the algorithm we use to extract fuzzy measures is the Bees algorithm, as it showed promise in previous work [71].

The predictor variables, as defined in [74], include high school class rank percentile, mathematics placement score, direct matriculation from high school, and intention to work.

The existing students data are from a set of first-time undergraduate students who enrolled in fall semester.

Using the above-described data, our goal is to show that: (1) our approach allows to recreate decisions (data) used to determine the reasoning process (fuzzy measure) with a higher accuracy than currently used techniques; and that (2) it also works to predict decisions (data) that were not included in determining the reasoning process (fuzzy measure) but that were available to us.

We were also interested in addressing the following questions:

1. Is students' academic performance consistent with the predictor variables? We wanted to know how well we would be able to predict students risk level according to their academic performance. For students with the same predictor variables, how different

the academic performance can be? Again, this was highly dependent on each student's individual characteristics and activities.

Experimental Results

The sample consisted 14,558 first-time, degree-seeking undergraduate students who entered the institution in 2003 to 2008 fall semesters. Students data were collected by the authors' institution's IR office, and the data source included the centralized database used by the institution and the survey data that had originally been used to gather information on intended work hours. Students without cumulative first-year GPA were excluded from the study. If students had first-year cumulative GPA, no matter they left the university or not, they were included in the study. For instance, John was first-time, degree-seeking undergraduate student who started in fall 2005. He completed all courses for fall 2005 and earned a 2.5 GPA. He didn't enroll in the following spring 2006 and summer 2006 semesters, meaning that he withdrew from the university for the rest two semesters of the academic year 2005-06. Therefore, his first-year cumulative GPA was actually his first-term GPA. Since he had first-year GPA, we included him in the study even he stayed only one semester at the university. If John withdrew all courses for fall 2005 and didn't enroll in spring 2006 and summer 2006, then he didn't earn any GPA points in his first year. He was excluded from the study since he didn't have first-year GPA.

We first extracted a fuzzy measure from all samples that fitted them all the best. In Table 6.5, we report the accuracy reached when determining the target fuzzy measure; i.e., the sum of the differences between the reconstructed data and the original data for all the data.In particular, we provide this information for different iteration counts of our Bees algorithm: we can observe that the quality is stabilized already after 100 iterations.

Although the results are stable after 100 iterations, it is about 20 times worse than the results of the toy examples in [71]. This is because students academic performance and the predictor variables are not consistent among different students. Especially, with the same predictor variables, the academic performance for different students may vary widely

Table 6.5: Mean Square Error for all samples

Iterations	**E**
100	0.1525677
1000	0.1525568

because the first-year GPA is highly dependent on each student's individual characteristics and activities.

The final obtained fuzzy measure using Bees algorithm is reported in Table 6.6, and we can observe that they are very close to what we expected to get.

Table 6.6: Obtained fuzzy measures using Bees algorithm

$\mu_1 = 0.5783053$	$\mu_{12} = 0.578383$	$\mu_{123} = 0.999838$
$\mu_2 = 0.5768046$	$\mu_{13} = 0.999838$	$\mu_{124} = 0.5784804$
$\mu_3 = 0.9997692$	$\mu_{14} = 0.5783289$	$\mu_{134} = 0.999838$
$\mu_4 = 0.4005329$	$\mu_{23} = 0.9997692$	$\mu_{234} = 0.9999663$
	$\mu_{24} = 0.5768046$	
	$\mu_{34} = 0.9997692$	

We observe that, among 4 criteria we used to calculate Choquet integral, that is, high-school class rank percentile, mathematics placement score, direct matriculation from high school, and intention to work, direct matriculation from high school plays a most important role in determining first-year academic performance. Although the importance of a coalition increased a little by including other factors that may also affect first-year academic performance, direct matriculation from high school clearly dominates the decision of first-year academic performance.

Analysis of our results

What we have shown in above is how well the Bees algorithm used to extract fuzzy measure. Now we want to evaluate how well the extracted fuzzy measure can help predict future students risk level according to their first-year academic performance.

We compared the evaluations we obtained to the original experts' decision and assessed the accuracy of our approach using 2 different evaluation processes.

1. Eval1: For each group, we calculated how close the predicted decisions (calculated using Choquet integral) are to the known experts' decisions using Gaussian distribution $f = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$. We compute a percentage of accuracy for each predicted decision as follows: the closer the predicted decision is to the average assessment, the higher the accuracy is.

2. Eval2: We used the same average as for Eval1 but allowed for some flexibility by accepting any evaluation within σ (standard deviation) of the target average. This evaluation accounted for the uncertainty of the experts' decisions. Every predicted decision between $\mu - \sigma$ and $\mu + \sigma$ is considered to be an accurate decision.

The overall evaluation results for all cohorts are in Table 6.7, and we also list the results using the Cox proportional hazards (PH) regression model to compare with.

Table 6.7: Accuracy using Bees algorithm

Cohorts	PH regression model [74]	Eval 1 (Average)	Eval 2 ($\mu \pm \sigma$)
All	38.14%	58.32%	57.29%

Discussion

This study focuses on classifying students according to their risk of departure from the college. The criteria we used are based on student information available at the time of

admission, such as, high school class rank percentile, mathematics placement score, direct matriculation from high school, and intention to work (see section 6.2.1 for detail). This early classification allows the institution to address students needs during the first year and to act immediately to reduce the possibility of departure. However, although the results of our study have shown much more accuracy than the existing approach (in 6.2.1), the actual accuracy of our approach is around 60%, which means 40 out of 100 students will be classified into a wrong category.

One possible reason of the low accuracy is that the decision is made according to students pre-college experience, which may change a lot during the college period. On the other hand, some other criteria, which we ignored, may also affect the students risk level; for example, gender, race and ethnicity, etc. In particular, since 1988, more females than males have been enrolled in the colleges, and by 2009, female enrollment is around 59 percent [2]. Research results have shown that the retention is affected by student's gender. For example, gender was significantly related to the retention in [54], with women are more likely to retain than men. When combined with different predictor variables, gender may play a different role with either more importance or less importance [54]. Race and ethnicity also affect the retention. Asian American and/or White students are more likely to be retained in college [54], and the under-represented minority (URM) students are less likely to be retained. More research must be conducted to determine the appropriate set of criteria to classify students risk levels.

Moreover, as we explained in section 3, for a set with n criteria, 2^n values of fuzzy measure are needed, that is, with the number of criteria increased, the number of variables in 3.3 increase exponentially. The more criteria we need to classify students, the more complicated the objective function is. Therefore, we need to find a tradeoff between the number of criteria and the complexity of the objective function.

Summary. In this chapter, we showed that our contributed algorithms could be successfully applied to two real problems with good results: software quality assessment and

at-risk students prediction.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this dissertation, we considered the problem of making decision under multiple, usually conflicting, criteria. Such decisions are hard to make and can be costly. We addressed the problem of automating the making of such decisions and we chose to solve so-called multi-criteria decision making problems through the use of fuzzy measures and integrals. We identified the specific problem of extraction of fuzzy measures and conducted our work in designing new extraction algorithms and in showing the applicability of our findings in real-world settings.

In our contribution to fuzzy measure extraction algorithms, we first proposed a Bees algorithm to extract fuzzy measures from sample decision data. The main advantages of using the Bees algorithm are that it performs a global search on the search space and that it reduces the risk of falling into local optima. In comparison with existing optimization algorithms that had previously been applied to FME, such as genetic algorithms and the gradient descent algorithm, our experimental results of the Bees algorithm showed that our Bees algorithm was faster and provided at least similar accuracy as or better than existing algorithms.

We also proposed an adaptive hybrid approach that combines the Bees algorithm with an interval constraint solver – RealPaver – that narrows down the search space to focus the Bees' search on areas that are likely to contain better values of the objective function.

Finally, another interval-based algorithm that we proposed speculates on the value of the objective function and always bets that the optimum value of the objective function

lies in the bottom part of the function's overestimated interval range. We then used an interval constraint solver, RealPaver, to verify our speculations on the value of the objective function. By using RealPaver, the search space is narrowed down and the global results can be guaranteed, as RealPaver always return a search space that contains all solutions that satisfy the constraints if there are solutions to the problem. By speculating on the range of the objective function, the search performance can be sped up, as those speculations whose solutions are not found by RealPaver are discarded, focusing on the speculated ranges for the objective function. In comparison with other FME algorithms, the results of speculative algorithms are closer to 0, showing a better quality of the results.

In this research, we also wanted to show that the algorithms we had developed and tested on toy examples were also relevant and efficient in real-world settings. Our objective was to obtain good results in predicting expert(s) decisions, and to possibly outperform existing automated techniques' results when available.

We tested the Bees algorithm and the hybrid interval-based algorithm in the context of software quality assessment (SQA). By viewing the SQA problem as a multi-criteria decision making (MCDM) problem, we were able to use fuzzy measures to model software expert's decision making process and help predict/evaluate software quality. We were able to show that our approaches (specifically fuzzy measure extraction based on experts' decisions data) helps to predict/evaluate software quality with consistently over 60% accuracy, which is as accurate as or (in most cases) better than previous approaches to SQA conducted using the traditional machine learning techniques.

We also proposed a multi-criteria-decision-making-based approach to predict students risk level according to their first-year academic performance. By viewing this prediction/classification problem as a multi-criteria decision making (MCDM) problem, we were able to use fuzzy measures to model students' performance and help predict/evaluate at-risk students. We were able to show that our approach (specifically fuzzy measure extraction based on experts' decisions data) helps to predict/evaluate students risk level with consistently close to 60% accuracy, which is more accurate than previous approaches to at-risk

students classification conducted using PH regression model.

7.2 Future Work

Our current approach can be further improved as follows. Although the hybrid algorithms we implemented provide good and reasonably fast results for fuzzy measure extraction, we believe we can improve them by designing better interaction modes and possibly involving a third solver in the process. Moreover, the experts' opinions (data used to extract a decision process model) usually are linguistic values; for example, Excellent, Good, Fair, Poor, and Bad. These words have different meanings to different experts, and therefore, experts' linguistic ratings are uncertain and their interpretation should not be uniform. In particular, using a continuous utility function that assigns a precise value to each of these evaluation results would result in lower accuracy. In order to better fit the experts' opinions, in the future, we will use an interval end-point approach [38] and may use non-linear utility functions. Finally, we plan to study and quantify the amount of data that is necessary to extract meaningful and accurate decision process models: for instance, what is the impact of a reduced sample data set on the quality of the decisions? What is the critical number of data w.r.t. the number of criteria?

References

[1] S. H. Alavi, J. Jassbi, P. J. A. Serra, and R. A. Ribeiro. Defining fuzzy measures: A comparative study with genetic and gradient descent algorithms. In *Intelligent Engineering Systems and Computational Cybernetics*, pages 427–437. Springer, 2009.

[2] S. Aud, W. Hussar, F. Johnson, G. Dena, E. Roth, E. Manning, X. Wang, and J. Zhang. *The Condition of Education 2012. NCES 2012-045.* National Center for Education Statistics, 2012.

[3] J. Bansiya and C. Davis. A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on Software Engineering*, 28(1):4–17, 2002.

[4] D. Bouyssou. Some remarks on the notion of compensation in (MCDM). *European Journal of Operational Research*, 26:150–160, 1986.

[5] P. M. Brewerton and L. J. Millward. *Organizational Research Methods: A Guide for Students and Researchers.* Sage Publications, 2001.

[6] L. Carlozo. Why college students stop short of a degree. `http://johngress.com/2012/03/27/why-college-students-stop-short-of-a-degree`.

[7] M. Ceberio and F. Modave. An interval-valued, 2-additive Choquet integral for multi-criteria decision making. In *Proceedings of the 10th Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'04)*, Perugia, Italy, July 2004.

[8] E. F. Combarro and P. Miranda. Identification of fuzzy measures from sample data with genetic algorithms. *Computers & Operations Research*, 33(10):3046–3066, 2006.

[9] A. F. Garcia Contreras, X. Wang, M. Ceberio, R. Bixler, and L. Gutierrez. Interval optimization to predict software quality assessment decisions. In *INFORMS Optimization Society Conference 2012*, Coral Gables, FL, February 2012.

[10] L. Davis. *Genetic Algorithms and Simulated Annealing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1987.

[11] L. M. de Campos and M. J. Bola *n*os. Characterization and comparison of Sugeno and Choquet integrals. *Fuzzy Sets and Systems*, 52:61–67, 1992.

[12] J. S. Dodgson, M. Spackman, A. Pearman, and L. D. Phillips. Multi-criteria analysis: a manual. Technical report, Department for Communities and Local Government: London, London, UK, 2009.

[13] E. Elbeltagi, T. Hegazy, and D. Grierson. Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Infomatics*, 19:43–53, 2005.

[14] L. H. Etzkorn, W. E. J. Hughes, and C. G. Davis. Automated reusability quality analysis of oo legacy software. *Information & Software Technology*, 43(5):295–308, 2001.

[15] Z. W. Geem, J. H. Kim, and G. V. Loganathan. A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2):60–68, 2001.

[16] M. Grabisch. A new algorithm for identifying fuzzy measures and its application to pattern recognition. In *Proceedings of 4th IEEE International Conference on Fuzzy Systems*, volume 1, pages 145–150, Yokohama, Japan, March 1995.

[17] M. Grabisch. The application of fuzzy integrals in multicriteria decision making. *European Journal Of Operational Research*, 89(3):445–456, 1996.

[18] M. Grabisch. Fuzzy integral for classification and feature extraction. In M. Grabisch, T. Murofushi, and M. Sugeno, editors, *Fuzzy Measures and Integrals: Theory and Applications*, pages 415–434. Physica Verlag, 2000.

[19] M. Grabisch, I. Kojadinovic, and P. Meyer. A review of methods for capacity identification in Choquet integral based multi-attribute utility theory applications of the Kappalab R package. *European Journal Of Operational Research*, 186:766–785, 2008.

[20] M. Grabisch, H. T. Nguyen, and E. A. Walker. *Fundamentals of uncertainty calculi with applications to fuzzy inference*. Kluwer Academic Publishers, Norwell, MA, 1994.

[21] M. Grabisch and M. Roubens. Application of the Choquet integral in multicriteria decision making. In M. Grabisch, T. Murofushi, and M. Sugeno, editors, *Fuzzy Measures and Integrals: Theory and Applications*, pages 348–374. Physica Verlag, 2000.

[22] L. Granvilliers and F. Benhamou. Realpaver: An interval solver using constraint satisfaction techniques. *ACM Transactions on Mathematical Software (TOMS)*, 32(1):138–156, 2006.

[23] A. Guitouni and J. Martel. Tentative guidelines to help choosing an approapriate (MCDA) method. *European Journal of Operational Research*, 109:501–521, 1998.

[24] M. Hassoun. *Fundamentals of Artificial Neural Networks*. MIT Press, 1995.

[25] S. D. Hosek, P. Tiemeyer, R. Kilburn, D. A. Strong, S. Ducksworth, and R. Ray. *Minority and Gender Differences in Officer Career Progression*. Rank Publishing, 2001.

[26] S. Hwang and R. He. Improving real-parameter genetic algorithm with simulated annealing for engineering problems. *Advances in Engineering Software*, 37(6):406–418, 2006.

[27] National Cancer Institute. Breast cancer treatment option overview. `http://www.cancer.gov/cancertopics/pdq/treatment/breast/Patient`. Visited December 2012.

[28] U. Jensen. Factors influencing student retention in higher education. Technical report, Kamehameha Schools Research & Evaluation, Honolulu, HI, 2011.

[29] C. Kahraman. Multi-criteria decision making methods and fuzzy sets. In C. Kahraman, editor, *Fuzzy Multi-Criteria Decision Making: Theory and Applications with Recent Developments*, pages 1–18. Springer, 2008.

[30] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Perth, Australia, 1995.

[31] S. Kirkpartick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[32] B. Kitchenham, S.Pfleeger, and N. Fenton. Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering*, 21(12):929–944, 1995.

[33] S. G. Komen. Treatment choices - an overview. `http://ww5.komen.org/uploadedFiles/Content_Binaries/806-386.pdf`. Visited December 2012.

[34] G. D. Kuh, J. Kinzie, J. Buckley, J.A. Bridges, and J. C. Hayek. What matters to student success: A review of the literature. *Commisioned Report for the National Symposium on Postsecondary Student Sucess: Spearheading a dialog on student success*, 2006.

[35] M. Lorenz and J. Kidd. *Object-oriented software metrics: a practical guide*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.

[36] T. Magoc and V. Kreinovich. How to relate fuzzy and OWA estimates. In *Proceedings of Annual Conference of North American Fuzzy Information Processing Society (NAFIPS'2010)*, Toronto, Canada, July 2010.

[37] M. Mathur, S. B. Karale, S. Priye, V. K. Jayaraman, and B. D. Kulkarni. Ant colony approach to continuous function optimization. *Industrial & Engineering Chemistry Research*, 39(10):3814–3822, 2000.

[38] J. Mendel. Computing with words and its relationships with fuzzistics. *Information Sciences*, 177:988–1006, 2007.

[39] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, first edition, 1997.

[40] F. Modave, M. Ceberio, and V. Kreinovich. Choquet integrals and OWA criteria as a natural (and optimal) next step after linear aggregation: A new general justification. In *Proceedings of MICAI'2008*, pages 741–753, 2008.

[41] F. Modave, D. Dubois, M. Grabisch, and H. Prade. A Choquet integral representation in multicriteria decision making. In *Proceedings of AAAI Fall Symposium*, Boston, MA, 1997.

[42] F. Modave and P. W. Eklund. A measurement theory perspective for MCDM. In *Proceedings of the 10th IEEE International Conference on Fuzzy Systems*, pages 1068–1071, Melbourne, Australia, 2001.

[43] D. L. Moody. Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering*, 55(3):243–276, 2005.

[44] T. Murofushi and M. Sugeno. Fuzzy measures and fuzzy integrals. In M. Grabisch, T. Murofushi, and M. Sugeno, editors, *Fuzzy Measures and Integrals: Theory and Applications*, pages 3–41. Physica Verlag, 2000.

[45] G. A. Norris and H. E. Marshall. Multiattribute decision analysis method for evaluating buildings and building systems. Technical report, US Department of Commerce National Institute of Standards and Technology, Gaithersburg, MD, September 1995.

[46] International Society on Multiple Criteria Decision Making. `http://www.mcdmsociety.org/`. Visited December 2012.

[47] J. Osbeck, S. Virani, O. Fuentes, and P. Roden. Investigation of automatic prediction of software quality. In *Proceedings of Annual Conference of North American Fuzzy Information Processing Society (NAFIPS'2011)*, El Paso, TX, March 2011.

[48] M. Parker. Placement, retention, and success: A longitudinal study of mathematics and retention. *The Journal of General Education*, 54(1):22–40, 2005.

[49] M. C. Paulk, C. V. Weber, B. Curtis, and M. B. Chrissis. *The capability maturity model: guidelines for improving the software process.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[50] D. Pham, A. Ghanbarzadeha, E. Koc, S. Otri, S. Rahim, and M. Zaidi. The bees algorithm – a novel tool for complex optimization problems. In *Proceedings of 2nd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2006)*, pages 454–459, 2006.

[51] R. Poli, J. Kennedy, and T. Blackwell. Particle swam optimization. *Swarm Intelligence*, 1(1):33–57, 2007.

[52] R. Pressman. *Software Engineering: A Practitioner's Approach.* McGraw-Hill, 2005.

[53] J. Raol and S. Mankame. Artificial neural networks - a brief introduction. *Resonance*, 1(2):47–54, 1996.

[54] R. D. Reason. Student variables that predict retention: Recent research and new developments. *Journal of Student Affairs Research and Practice*, 46(3):482–501, 2009.

[55] R. Sedgewick and K. Wayne. *Introduction to Programming in Java: An Interdisciplinary Approach.* Addison-Wesley, 2007.

[56] PassMark Software. CPU benchmarks.
http://www.cpubenchmark.net/.

[57] I. Sommerville. *Software Engineering*. Addison Wesley Publishing Company, Harlow, England, 2004.

[58] L. S. Stratton, D. M. O'Toole, and J. N. Wetzel. Are the factors affecting dropout behavior related to initial enrollment intensity for college undergraduates? *Research in Higher Education*, 48(4):453–485, 2007.

[59] L. S. Stratton, D. M. O'Toole, and J. N. Wetzel. A multinomial logit model of college stopout and dropout behavior. *Economics of Education Review*, 27(3):319–331, 2008.

[60] E. Takahagi. Usage: Fuzzy measure-Choquet integral calculation system (λ fuzzy measure and sensitivity analysis).
http://www.isc.senshu-u.ac.jp/~thc0456/Efuzzyweb/mant2/mant2.html.

[61] E. Takahagi. A fuzzy measure identification method by diamond pairwise comparisons and ϕ_s transformation. *Fuzzy Optimization and Decision Making*, 7(3):219–232, 2008.

[62] A. Toloie-Eshlaghy and M. Homayonfar. MCDM methodologies and applications: A literature review from 1999 to 2009. *Research Journal of International Studies*, 21:86–137, 2011.

[63] S. S. Virani, S. Messimer, P. Roden, and L. Etzkorn. Software quality management tool for engineering managers. In *Proceedings of the Industrial Engineering Research Conference*, pages 1401–1406, Vancouver, Canada, 2008.

[64] J. Wang and Z. Wang. Using neural networks to determine Sugeno measures by statistics. *Neural Networks*, 10(1):183–195, 1997.

[65] W. Wang, Z. Wang, and G. J. Klir. Genetic algorithms for determining fuzzy measures from data. *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, 6(2):171–183, 1998.

[66] X. Wang and M. Ceberio. Fuzzy measure extraction for predicting at-risk students. In *Proceedings of 2nd World Conference on Soft Computing*, Baku, Azerbaijan, December 2012.

[67] X. Wang, M. Ceberio, S. Virani, A. F. Garcia Contreras, and J. Cummins. A hybrid algorithm to extract fuzzy measures for software quality assessment. *Journal of Uncertain Systems*, 2012. Submitted for publication.

[68] X. Wang, M. Ceberio, S. Virani, C. Del Hoyo, and L. Gutierrez. Fuzzy measure extraction for software quality assessment as a multi-criteria decision-making problem. In *Proceedings of The 2012 World Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP'12)*, Las Vegas, NV, July 2012.

[69] X. Wang, A. F. Garcia Contreras, M. Ceberio, C. Del Hoyo, and L. C. Gutierrez. A speculative algorithm to extract fuzzy measures from sample data. In *Proceedings of WCCI 2012 IEEE World Congress on Computational Intelligence*, Brisbane, Australia, June 2012.

[70] X. Wang, A. F. Garcia Contreras, M. Ceberio, C. Del Hoyo, L. C. Gutierrez, and S. Virani. Interval-based algorithms to extract fuzzy measures for software quality assessment. In *Proceedings of Annual Conference of North American Fuzzy Information Processing Society (NAFIPS'2012)*, Berkeley, CA, August 2012.

[71] X. Wang, J. Cummins, and M. Ceberio. The Bees algorithm to extract fuzzy measures for sample data. In *Proceedings of Annual Conference of North American Fuzzy Information Processing Society (NAFIPS'2011)*, El Paso, TX, March 2011.

[72] Z. Wang, K. Leung, and J. Wang. A genetic algorithm for determing nonadditive set functions in information fusion. *Fuzzy Sets and Systems*, 102(3):463–469, 1999.

[73] J. W. Weiss and D. Anderson Jr. CIOs and IT professionals as change agents, risk and stakeholder managers: A field study. In *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)*, volume 8, page 249.3, 2003.

[74] B. T. Weldeslassie, R. Mathew, L. A. Schmersal, D. Carrejo, and D. Herring. Identifying students at risk of departure from college (unpublished manuscript). 2009.

[75] K. Yoon and C. Hwang. *Multiple Attribute Decision Making: an Introduction*. Sage Publications, Thousand Oaks, CA, 1995.

Curriculum Vitae

Xiaojing Wang was born in Nanjing, Jiangsu Province, China, the youngest child of Yushu Wang and Feng Xu. She entered Nanjing University of Technology, Nanjing, China in 1992 and received a Bachelor's degree in Bioengineering in 1996. In 2001, she entered the University of Texas at El Paso, Texas, pursuing a master's degree in Computer Science and received her M.S. degree in 2004. To further advance her research interest she started her doctoral study in University of Texas at El Paso in Spring, 2005. While pursuing the doctoral degree at UTEP, she worked as teaching assistant in the Computer Science Department in 2005-2007. In Summer 2007, she began working full time as a researcher at UTEP's Center for Institutional Evaluation, Research and Planning (CIERP).

Permanent address: 5-507 Lanjiazhuang
Nanjing, 210018
P. R. China