Handling imperfect information in practical applications.

Robin De Mol

August 20, 2019

Contents

Korte samenvatting

We zitten momenteel midden in een heuse revolutie waarin de computer zijn opmars maakt. Op steeds meer plaatsen zien we dat computers en robots voorgaande systemen vervangen, gaande van gebieden op vlak van fysieke arbeid tot archivering in de vorm van dataopslag en dataverwerking. Het is dankzij de computer dat we vandaag grotere hoeveelheden informatie kunnen verwerken aan hogere snelheden dan ooit tevoren. Toch merken we dat computers nog steeds niet in staat zijn om zelfstandig intelligent te werken. Ondanks hun enorme rekenkracht, opslagcapaciteit, verwerkingssnelheid en verbondenheid zijn computers slechts werktuigen.

Het valt op dat computers op een volkomen andere manier omgaan met informatie dan mensen. Voor een computer bestaat informatie uit duidelijk gestructureerde, exacte datapunten. Mensen, daarentegen, zijn het gewoon om te werken met imperfecte gegevens. Dit is grotendeels te wijten aan het vermogen van de mens om te communiceren, denken en redeneren in natuurlijke taal. Men moet bij het autorijden niet exact weten hoe snel men rijdt en hoe ver men zich van een rood licht bevindt om op tijd tot stilstand te komen. Het volstaat dat men kan inschatten dat men "snel" gaat en zich "dichtbij" bevindt om te weten dat men "hard" moet remmen. Het imperfect zijn van gegevens belet de mens dus niet om te functioneren. In tegendeel, het is dankzij de flexibiliteit van het menselijk redeneervermogen om te redeneren en handelen basis van imperfecte informatie dat men in staat is om complexe taken, zoals autorijden, veilig uit te voeren.

Op dit vlak verschilt de computer duidelijk van de mens. Voor een computer bestaat informatie uit een verzameling datapunten. Elk datapunt is in essentie een sequentie van bits, die geïnterpreteerd kan worden als een getal of als een letter. Het is mede te danken aan de focus van de exacte wetenschappen op precieze en zekere resultaten dat men zich ook bij de ontwikkeling van de computer geconcentreerd heeft op systemen die uitgaan van perfecte informatie uit een ideale wereld.

De ideale wereld is echter niet de reële wereld. Bijgevolg is men nu op zoek naar manieren om computers verder te laten evolueren zodat deze in staat worden gesteld om imperfecte informatie op een meer natuurgetrouwe manier te beheren en verwerken. Computers volledig zelfstandig laten redeneren blijkt bijzonder moeilijk te zijn. Zelfs al zou een computer kunnen redeneren, dan nog is de manier waarop deze data opslaat fundamenteel verschillend van de

manier waarop mensen met data omspringen. Zo slaat een computer bijvoorbeeld geen context op waarin de informatie wordt verkregen. Computers houden ook geen rekening met eenheden. Hierdoor is het perfect denkbaar om met een computer analyses uit te voeren die absoluut geen steek houden. Men kan bijvoorbeeld leeftijden en gewichten bij elkaar optellen. Terwijl een redelijk mens dit nooit zou doen, zal een computer hier niet moeilijk over doen. Het is dus nog steeds aan de mens om bedachtzaam en verantwoordelijk om te springen met het gebruik van de computer. Bijgevolg is de brug tussen de computer en de menselijke gebruiker van groot belang, want deze bepaalt namelijk de mate waarin de resultaten van analyses zinvol zijn.

De vaagverzamelingenleer introduceert een theoretisch, wiskundig raamwerk dat aan de basis ligt van vaaglogica en possibiliteitstheorie. Vaaglogica vormt een formele basis om om te gaan met vage en onnauwkeurige informatie, terwijl possibiliteitstheorie kan worden gebruikt om om te gaan met onzekere informatie. Dit werk situeert zich in het onderzoeksgebied dat deze theorie probeert te integreren in computersystemen. Hierbij zijn er nog steeds verscheidene uitdagingen, waarvan we er een aantal in detail bestuderen in deze thesis:

- Hoewel het duidelijk is hoe we kunnen omgaan met vage informatie enerzijds en onzekere informatie anderzijds, is het nog niet volledig duidelijk hoe we kunnen omgaan met de combinatie van de twee. We beschouwen dit in de context van flexibele, vage bevraging op databanken die onzekere data bevatten. Het onderzoek uit dit doctoraat heeft geleid tot de introductie van een nieuwe modelling van de resultaten van een dergelijke bevraging, waarmee zowel vaagheid als onzekerheid samen voorgesteld kunnen worden. Dit deel van het werk is eerder theoretisch van aard, aangezien men in feite de fundamentele manier waarop computers data opslaan zou moeten aanpassen. Er wordt echter ook een brug gelegd naar de realiteit door een aantal restricties op te leggen, waardoor het mogelijk wordt om een vereenvoudigd model te realiseren dat bruikbaar is met de computers van vandaag.

- Hoewel het een mooi resultaat is om een manier te hebben om het resultaat van een vage bevraging op onzekere data voor te stellen, zijn praktische beslissingen typisch niet gebaseerd op één enkel criterium. Er is een hele onderzoekstak gewijd aan beslissinsondersteuningssystemen, de studie van hoe men, uit een aantal alternatieven, besluit om een bepaalde keuze te maken. Hierbij wordt gekeken naar hoe men een algemeen besluit kan vormen over een alternatief op basis van verschillende eigenschappen. Dit wordt typisch gedaan door criteria op te stellen, te evalueren en de resultaten van individuele criteria te aggregeren met elkaar tot één geheel. Ook dit proces moet worden herzien in het kader van het werken met imperfecte gegevens. Het zal blijken dat de introductie van de nieuwe modellering nieuwe opties met zich meebrengt. We bestuderen onder andere de mogelijkheid om ons nieuw model rechtstreeks te gebruiken, maar ook de keuze om ze om te zetten in een ander formaat dat compatibel

is met huidige technieken. We stellen ook een nieuwe aggregatietechniek voor en onderzoeken hoe deze praktisch gebruikt kan worden.

- Computers maken gebruik van indices om efficiënt te kunnen zoeken naar gevraagde gegevens. Klassieke technieken om dit te verwezenlijken steunen op het exact gekend zijn van gegevens. Wanneer we toestaan dat informatie onzeker is, kunnen deze technieken niet langer worden toegepast. We moeten dus op zoek gaan naar manieren om indices te bouwen voor onzekere informatie. In dit werk introduceren we een nieuwe uitbreiding van een klassieke B^+-tree indexeringstechniek, specifiek om onzekere informatie te kunnen indexeren. Het zal blijken dat het mogelijk is om een efficiënte index aan te maken, maar dit onderzoek leidt ook tot een aantal interessante observaties die een verband constateren tussen de mate van onzekerheid en performantie. We bestuderen deze verbanden, bespreken waar het eventueel nog mogelijk is om verder te optimaliseren, en positioneren onze resultaten ten opzichte van andere onderzoeksresultaten in hetzelfde domein.

Dit werk draagt bij tot het onderzoek naar het verwerken van imperfecte gegevens met computers. Dit heeft tal val mogelijke toepassingsgebieden, zoals onder andere beslissingsondersteuningssystemen, sorteersystemen, ruimtelijke modellering, aanradingssystemen, geautomatiseerd dataherstel, data kwaliteit, regelsystemen, en zo voort. De focus ligt hierbij vooral op het waarheidsgetrouw voorstellen van informatie, met zo min mogelijk verlies van informatie door te aanvaarden dat deze imperfect kan zijn.

Short summary

We are currently in the middle of a serious revolution in which the computer is slowly invading every corner of the world. In increasingly more areas, computers are replacing previous systems, going from physical labor to data storage and processing. Thanks to computers, we are now able to process larger amounts of information at higher speeds than ever before. Still, we notice that computers are not yet capable of independent, intelligent thinking. Despite their enormous computing power, storage capacity, processing speed and interconnectivity, computers are only tools.

It is noticeable that computers treat information in a completely different way than humans do. For a computer, information consists of precisely structured, exact points of data. Humans, however, are used to dealing with imperfect information. This is largely because of the human capability of communicating, thinking and reasoning in natural languages. One does not need to know exactly how fast they are travelling and how far away from a stopping light they are in order to come to a standstill in time. It suffices to know that one is going "fast" and that one is "close" to the crossroads in order to decide that one should brake "hard". Data imperfections do not hinder people from functioning. On the contrary, it is due to the flexibility of being able to process imperfect information, that people are capable of performing complex tasks, like driving a car, in a safe manner.

Computers are very different from people in this regard. For a computer, information is stored as a collection of points of data. Each point of data is essentially just a sequence of bits that can be interpreted as a number or a character. This is partly due to the focus of exact science research on precise and certain results. During the development of computers, this has lead to a focus on systems that assume perfect working conditions, i.e., perfect information in an ideal world.

The real world is, however, far from ideal. This is why researchers are working on ways to evolve computers in such a way that they become capable of working with realistic, imperfect information. Enabling computers to reason independently turns out to be extremely difficult. Even if computers were able to reason autonomously, the way they store information remains fundamentally different from the way people do. A computer, for example, is not aware of the context in which it receives information. They also do not store the units

in which data are expressed. Because of this, it is perfectly possible to use computers to perform analyses that produce nonsensical results. One could add up ages and weights, for instance. Though a human would never do this, a computer would not complain when tasked to do so. It is the responsibility of the user to act responsibly and to reinvent the context that was lost when information was digitized. Consequently, the bridge between the computer and the user is of great importance, as it plays an important role in the degree to which the user is able to model their behavior and obtain results that are intuitive.

Fuzzy set theory introduces a theoretical, mathematical framework that forms the foundation of both fuzzy logic and possibility theory. Fuzzy logic is a formal base for dealing with fuzzy and imprecise information, while possibility theory can be used to deal with uncertainty. This work is situated in the research area that focusses on trying to implement this theory on computers. There are plenty of challenges in this area, from which we will study a few in detail:

- Though it is clear how we can deal with fuzzy information on the one hand and with uncertain information on the other, it is not yet obvious how both can be combined. We study this problem in the context of flexible querying on uncertain databases. This research has lead to the introduction of a new representation of the results of such an evaluation, which is capable of representing both fuzziness and uncertainty simultaneously. This work is mostly of a theoretic nature, since the fundamental way of how computers store data would have to be changed for the theory to be applicable in practice. However, a connection to today's reality is made by working under certain restrictions, simplifying the problem sufficiently such that it becomes realizable with discrete, numerical machines.

- Though it is nice to have a way to represent the result of a flexible query on uncertain data, practical decisions are typically based on more than a single criterion. An entire branch of research is dedicated to decision theory, the study in which it is investigated how and why people choose for particular solutions from a set of alternatives. This includes how a conclusion is made regarding a single alternative based on multiple different attributes. This is typically done by evaluting different attributes individually and aggregating the results thereof. This process needs to be revisited when dealing with imperfect information. As it turns out, introducing the new representation for query evaluation results raises new questions. We will investigate the option to aggregate these new representations directly, but also the option to transform them into a different format that is more compatible with current techniques. We will even take a short side step and reimplement an existing aggregation operator in a different aggregation framework.

- Computers use indices in order to be able to look up information efficiently. Classical indexing techniques rely heavily on the fact that attribute val-

ues are precisely known. When imperfect information is considered, these techniques can obviously no longer be applied. We need to find new ways to create indices for uncertain information. In this work, we introduce a novel adaptation to a classical technique, specifically for indexing uncertain information. It turns out that it is possible to make an efficient index. However, this research also raises new questions regarding the impact of uncertainty on the performance of this technique. We discuss which properties have which influence and describe which areas can be further optimized.

This work contributes to the research area regarding the processing of imperfect information using computers. There are plenty of application areas, among which decision support systems, ranking systems, spacial modelling, recommender systems, automated data healing, data quality, controllers, and so on. The focus is mostly on representing information truthfully, minimizing any loss of information by not making the assumption that data are perfect.

Chapter 1

Introduction

As suggested by the title, this thesis will discuss some of the challenges regarding data quality in the ever-growing world of digital applications. More precisely, we will take a look at techniques for storing and querying uncertain data. Our focus will be on digital data, a fairly modern research area that has been growing at an incredible rate since the start of the digital revolution.

1.1 About data

In today's reports, you will often hear about "big data". This (perhaps misleading) umbrella term covers the many actual challenges in the digital data world. A rough classification in four categories (the four V's) provides a clearer understanding:

1. **Volume** With the explosion of devices that flood the market, more and more data are generated each day. Efficient handling of huge data volumes, often called "big" data, poses new challenges.

2. **Velocity** Not only the rate at which data are generated, but also the speed with which they can be analyzed. Being able to process "fast" data in due time (often real time) is another main challenge.

3. **Variety** Different sources of data do not necessarily (or likely) use the same representation or format. It is difficult to store "varied" data using fixed database schemes that require complex data transformations.

4. **Veracity** The results of data analytics can only be as good as the quality of the underlying data. Assuming that not all data are perfect, proper handling and reflecting the veracity of "bad" data is another challenge.

Of these categories, this thesis concentrates on problems related to veracity. Scrutinizing data quality is gaining popularity and the effects of negligence are becoming more and more tangible with the big data shift, but the subject is

a difficult one to approach. Data quality is an active research area but as a fairly broad and complicated topic, it is plagued by a lack of consensus, reflected by the many different published approaches for quantizing and dealing with uncertainty. Most approaches originate from a practical use case and formulate a solution that fits specific needs. Our research takes a more theoretical approach, focussing heavily on semantics and driven by questions like "What is data quality?", "How can it be measured?" and "What implications does uncertainty have on interpretability and usability?". Before we dive into mathematical definitions and formulas, we will have a short philosophical discourse on the definitions underlying data quality.

1.2 Measurements and information

Our senses continuously *measure* our surroundings, providing *information* to our brains. Arguably, one of the reasons why mankind has already evolved so far is because we share information with each other. Furthermore, we derive *knowledge* from our own experiences and pass it on to next generations. Evidently, we do so by communicating. Cave art is testament to the fact that expressing our thoughts is one of the first skills humans naturally developed. Our ways of communicating have matured from images to words, but also numbers play a fundamental role in exchanging information, particularly when describing measurements. Let us say a *datum* is the registration of a measurement, in essence a translation of information to symbols. A datum is typically represented by the combination of a value and a unit. For instance, if we are describing an object, it does not suffice to provide these values: 5, 20, 7, 1.5, 0, 0, 255. However, if we say its height is 5 cm, its width is 7 cm, its length is 20 cm, its weigth is 1.5 kg and its color is red, we can start to imagine this object: it could be a brick.

Besides our natural senses, people have developed many artificial "sensors" to measure a plethora of phenomena at varying levels of detail, some extremely accurate. In what follows, we will use the word "sensor" to denote a mechanism, natural or artificial, for measuring information. Take for example a clock. Essentially, it just shows numbers (data), but we know that these should be interpreted as indicators of time. To us, these numbers convey information. Note that this is so common that often the units are typically omitted, as they are implied to be understood.

1.3 The precision problem

Let us briefly reflect on the observation that sensors (natural or otherwise) always have an innate *precision*. Though discrete quantities can be measured with absolute precision, many physical quantities (such as time, temperature, length, weight, force, and so on) are typically treated as being continuous. Trying to state precisely what time it is, is impossible, because by the time your thought is finished, more time has passed. Even discrete quantities can be imprecisely spec-

ified, for numerous different reasons: an address may be only partially specified, a property can be described with a subjective term (e.g. movie-genre comedy) or there can be uncertainty regarding which particular discrete value was actual (e.g. a witness remembers a car but is unsure whether its color was black or blue).

Typically, sensors of higher precision are harder to construct and are more expensive as a result. Generally, however, we don't need high precision in our day to day lives: we don't need an atom clock in order to arrive at work in time, we don't need to know the exact temperature outside in order to dress accordingly, and so on. In fact, we often don't even need numerical data and have coded particular ranges of values into natural languages in the form of words such as "morning", "afternoon", "small", "heavy", "cheap", and so on. In his paper titled 'Computing with words' [10], Zadeh illustrates this by giving the example of driving a car: people can perform advanced tasks like performing a parallel parking manoeuvre, without any exact information and based purely on fuzzy readings of their environments such as estimated distances and speed. Though imprecise information does not allow (precise) calculations, it is often sufficient in order to be able to be useful in practice.

When using words to describe values, however, this hinges on the fact that there exists a common background knowledge, shared by everyone involved in the conversation, regarding the subject being discussed. Clearly, "cheap" will have different interpretations in the sentences "the new car I bought was cheap" and "we went to a cheap restaurant to celebrate my mother-in-law's birthday". "Cheap" is thus not defined as a single representation of a numeric range; its definition depends on the context in which it is used.

The take-away is that using sensors to measure things always implies a certain precision of information and that the application dictates which precision is required. As such, typically the measuring equipment is chosen in function of the application.

1.4 In a digital world

It is hard to imagine today's world without computers. From their broad spectrum of applications, we will focus on their capability to store, manage and process data. This bring us to the theory of databases: digital, persistent data stores. The question we want to answer is "how accurately can we store a piece of information in a database".

1.4.1 Computer language: a binary code

Databases come in different flavors, but how different they may be, at the lowest level they all express data in the same *binary* representation. Because this representation consists only of two symbols, 0 and 1, it is infeasible to rely solely on symbol sequences (*bitstrings*) in order to represent all possible values. Instead, a different approach which allows the re-use of bitstrings with different

semantics was devised. The result was the introduction of data types, such as INTEGER and CHARACTER. A data type essentially describes how a bitstring translates to a representation using other symbols.

The INTEGER data type is used to translate between bitstrings and integer numbers. The translation process is in fact just a translation from a base 10 to a base 2 representation of a numeric value, but the first digit is treated differently from the rest in order to be able to represent the signum (positive or negative).

Bitstrings can also be mapped to the characters of the different alphabets of the world with the corresponding encodings, like ISO 8859-1 for Western European, GB 2312 for Chinese, KS X 1001 for Korean, and so on. Essentially, these encodings are standardized, fixed look-up tables.

The digital representation of a datum thus consists of two parts: a bitstring and a data type. For example, the bitstring 01101100 can be used to represent the integer value 108, but also to represent the UTF-8 character 'k'. Note that there is no indication of precision. There is, however, in all cases, support for annotating data with a text-based label (e.g. *age*, *name*...). Chosen wisely, database designers can employ this to convey semantics and context but this does not change the fact that the database system does not fully understand this information. As such, the responsibility of using the data correctly is placed with both the creators and the users of the database. Though this is not necessarily an issue in an ideal world, the real world has proven that people do not always practice the expected discipline (like when a Mars lander crashed due to a mix up between the metric and the imperial system).

Despite their flaws, databases are used in almost every modern computer application.

1.4.2 Database systems for the handling of imperfect data

Many researchers have spent many hours thinking about how databases could be adapted such that these issues could be overcome. The many solutions can be roughly summarized under the umbrella term *fuzzy* databases, describing database systems for the handling of imperfect data.

A fuzzy database is a persistent data store that offers some degree of support for translating data to more than a bitstring and a data type. The most well-known solution is that of NULL (also ω or $\perp$) [1, 2, 5]. First proposed by E. Codd, the inventor of the relational database model, NULL is a unique symbol to represent that a certain piece of data is missing. NULL was introduced in the context of relational databases and is special because it has no data type yet can be used everywhere. Though it is a step in the right direction, NULL is not without controversy [6, 4, 3], as its addition forces users to employ three-valued logic.

NULL and three-valued logic

According to Boolean logic, statements are either TRUE or FALSE. In reality, however, we are sometimes faced with a situation wherein it is not possible to

evaluate which case is actual. There are several reasons why this can occur [7]:

- A measurement is of a lower level of precision than is required to plot a course of action. It is said to be *imprecise*. *Vagueness* is a special form of imprecision caused by communicating information using natural language (it's "hot" outside).
- No measurement has been performed (yet). It is said to be *missing*.
- It can not be measured because it does not exist (e.g. color of car for someone who does not own a car). It is said to be *inapplicable*.
- Different measurements contradict each other. They are said to be *inconsistent*.

These cases describe situations of *data imperfections* which make it impossible to say whether specific statements are TRUE or FALSE. It is said that the data are *uncertain*, though this is actually not strictly correct. It would be more accurate to say that there is uncertainty regarding the truthfulness of statements about the data, which may very well be perfectly known but simply not accurate enough to evaluate particular statements.

Three-valued logic adds UNKNOWN to the set of valid evaluation results regarding statements. This has far-reaching consequences, because it plunges us into a framework of three-valued logic. Consequently, the truth tables used to evaluate composite expressions must be extended, as shown in Table 1.1. As a result of doing so, expressions like $P \vee \neg P$ (P or not P) do not always evaluate to true [6], which is untuitive.

A	NOT(A)
F	T
U	U
T	F

AND	F	U	T
F	F	F	F
U	F	U	U
T	F	U	T

OR	F	U	T
F	F	U	T
U	U	U	T
T	F	T	T

Table 1.1: Extended truth tables for three-valued logic

A final remark on NULL that should not go unsaid is that it is simply not expressive enough: it does not yield sufficient flexibility to distinguish between inapplicable, missing or imprecise data. Nevertheless, NULL is widely used in databases due to its simplicity and added benefits outweighing its complications.

Set-based logic

A more general approach to accurately modelling data imperfections is based on set mathematics. The idea here is to represent information regarding an attribute by a set of values rather than a single, precise value. This adds more flexibility than NULL:

1. precise information is denoted by a singleton set

2. lack of information is reflected by the domain set (the set containing all possible values)

3. imprecise knowledge is represented by any other set (a strict subset of the domain set)

Though it is not a common convention, inapplicability could be represented by an empty set. Doing so would imply that the knowledge of the domain is complete (i.e. there are no possible values that are simply not known yet) so that the empty set can not be interpreted as "there could exist another, unknown but possible value". A more common solution is to extend each attribute's domain with a special NULL value, which should be interpreted as "inapplicable". This solution, though less practical as it re-introduces NULL with all of its downsides, is usually preferred because it offers more flexibility, making it possible to represent uncertainty about inapplicability itself (e.g.: "John's car is either black or John does not have a car").

This representation might seem bloated, but it is particularly interesting for numeric attributes (or more generally attributes which can be measured on an ordinal scale) because in those cases, sets of "adjacent" values correspond to *intervals*, which are well understood and which can be concisely described by its bounds. As it turns out, it is common to find sets of adjacent values, especially when representing imprecise information.

Recall that intervals also correspond to the natural way we use to deal with the precision problems regarding continuous quantities.

Indeed, by storing an interval for an attribute, we can reflect that our actual information is of a lower precision than that assumed by the user who provided the data. Regardless, "the precision assumed by the data provider" is likely undefined in which case it must be inferred by the data consumer.

For attributes measured on interval scales and beyond, we can even perform meaningful operations on multiple values, like comparing their length (not to be confused with size) and expressing the distance between them. We must, however, remain vigilant not to apply these operations on ordinal attributes, as in those cases the interval notation is merely a shorthand for a set, as is clear for nominal data.

Fuzzy set-based models

There are forms of information that can not be represented by mere sets. This is particularly true in areas concerning stochastic modelling, where there is evidence to suggest that certain values are more (or less) likely than others. In order to reflect this, each domain value should be mapped to a *degree* indicative of the amount of evidence for this value. Fuzzy set theory [8] is a generic framework for introducing grades in traditional systems. As such, the previously described set-based approach becomes a fuzzy set-based approach, where each domain value belongs to the fuzzy set to a certain degree, called the *grade of membership*. Typically, grades of membership are limited to the unit interval. A membership grade of 0 implies that the associated domain value is not part of

the set (as in the traditional definition) and that 1 implies maixmal membership. The closer a value's membership grade to 1, the more evidence there should be to believe that that value is the true value. Values can be compared to each other in the sense that a value with a higher membership grade is considered more believable. Note that fuzzy sets are a strict superset of regular sets, as a regular set can be implemented by mapping each value to a degree of either 0 or 1.

One of the several applications of fuzzy set theory is possibility theory [9], which is useful for modelling data imperfections.

The added richness of fuzzy sets comes at the price of another increase in complexity. Having to store a membership degree for each value could impose enormous storage requirements, and we can not use an interval to summarize a fuzzy set. Under the right circumstances, however, there is a way to summarize a fuzzy set with as little as four domain values. It will be shown that these circumstances are actually not that restrictive and that real-world problems often do not contradict the made assumptions. This and because it can be shown that cases where these circumstances do not apply can be decomposed into subcases where they do, motivates a strong focus on these interesting special fuzzy sets, sometimes referred to as fuzzy intervals.

1.5 Challenges and research topics

In this introduction, we have discussed the semantics of data and information. We have shown that possible improvements can still be made regarding how data are stored in and processed by existing database systems. This defines the main research question that has fueled our research: how to deal with uncertainty in the results of evaluating flexible queries on uncertain data?

It is not hard to argue that existing database systems could be improved upon, but it's a lot harder to actually implement changes. This is probably largely due to the fact that there are still too many unanswered questions regarding fuzzy set-based solutions. Additionally, existing solutions for concrete problems might have to be re-examined and possibly partially reinvented in a fuzzy way in order to be able to migrate existing systems to a new generation of computer intelligence, implying a certain amount of unavoidable downtime, which is undesirable for competing businesses. Examples of such solutions that must be revisited include fundamental operations such as indexing, query evaluation and query result interpretation (decision support), visualization, sorting and aggregation. In this thesis, we will go over some of these topics, give an overview of what has already been researched and introduce some novel contributions. More precisely, we will try to find an answer to the following research questions:

1. How should the evaluation results of a flexible query on uncertain data be represented *truthfully*?

2. What is the impact of incorporating uncertain data and flexible querying

on multi-criteria decision making (more precisely: on aggregation techniques)?

3. Today's systems are fast, and this is partly due to indexing techniques that rely on precise data. Working with uncertain data prevents the application of traditional indexing techniques. How can we adapt traditional indexing techniques so that they can be applied on uncertain data?

The remainder of this thesis is structured as follows. Chapter 2 contains some preliminaries regarding data modelling, database systems, fuzzy set theory and decision support that form the bedrock of the chapters that follow.

In Chapter 3, the impact of using fuzzy sets in both querying and data modelling is studied and the concept of a suitability distribution is introduced. In this chapter, the evaluation of flexible (or fuzzy) criteria on uncertain data is investigated.

Chapter 4 deals with decision support, a common application built on the added value of using fuzzy sets in querying. More precisely, it is examined how suitability distributions, which should be used when evaluating uncertain data, can be employed by extending existing techniques. Whereas Chapter 3 covers the evaluation of a single fuzzy criterion on a single uncertain datum, this chapter describes how the evaluation results of multiple criteria can be aggregated, especially when working with suitability distributions.

Chapter 5 covers the indexing of uncertain attribute values and defines a new data structure, the interval B^+-tree. This data structure is particularly fit for indexing interval-based values, and as such, uncertain data.

Finally, Chapter 6 summarizes the contributions that were presented in this dissertation and reflects on where further research is required.

Bibliography

[1] Edgar F Codd. "A relational model of data for large shared data banks". In: *Communications of the ACM* 13.6 (1970), pp. 377–387.

[2] Edgar F Codd. "Extending the database relational model to capture more meaning". In: *ACM Transactions on Database Systems (TODS)* 4.4 (1979), pp. 397–434.

[3] Chris J Date. *Relational Database Writings, 1994-1997*. Vol. 4. Addison-Wesley Longman, 1998.

[4] Christopher John Date and CJ Date. *Relational database writings, 1991-1994*. Vol. 4. Addison-Wesley Reading, MA, 1995.

[5] CJ Date. "Null Values in Database Management." In: *BNCOD*. 1982, pp. 147–166.

[6] CJ Date. *Relational database writings, 1985-1989, Volume 1*. Addison-Wesley, 1990, 1990.

[7] A Motro. *Modern database systems: the object model, interoperability, and beyond*. ACM Press/Addison-Wesley Publishing Co., 1995.

[8] L.a. Zadeh. "Fuzzy sets". In: *Information and Control* 8.3 (1965), pp. 338–353. ISSN: 00199958. DOI: 10.1016/S0019-9958(65)90241-X.

[9] Lotfi Zadeh. "Fuzzy Sets as a Basis for Possibility". In: *Fuzzy Sets and Systems* 1 (1978), pp. 3–28.

[10] Lotfi A Zadeh. "Fuzzy logic= computing with words". In: *Computing with Words in Information/Intelligent Systems 1*. Springer, 1999, pp. 3–23.

Chapter 2

Preliminaries

In this chapter, we will briefly cover the technologies that are fundamental to the remaining chapters and introduce symbols for frequently recurring terms. The topics that will be covered are databases and data modelling (2.1), fuzzy set theory (2.2) and aggregation (2.3).

2.1 Databases and entity-relationship modelling

Databases are first and foremost digital data stores. Practically every application uses a database, but as different applications have different needs, different database models have been devised. While database models differ in how they logically organize data, they are similar at a conceptual level.

When setting up a new database, it is good practice to start by making an entity-relationship (ER) model. This data modelling technique translates the description of the information that has to be stored into a collection of *entities* and *relationships*. In fact, the term 'entity' is poorly chosen, as ER models actually describe entity *types* instead of particular entities, which represent the data.

2.1.1 Entity types

An entity type can be treated as a *signature* $\mathcal{R}$ that defines a specific, finite set of attributes (or properties, characteristics, dimensions). Each attribute is denoted by a name a and its domain A, which specifies all legal values a can take. Usually, a domain is directly represented by a data type (Boolean, integer number, real number, text value, enumeration). Any restrictions on domains that can be derived from data types must be specified. Entities r themselves are instances of a signature, essentially corresponding to an assignment of attribute values v for each attribute from their respective domains. The collection of all entities of the same entity type with signature $\mathcal{R}$ is referred to as the *body* $\mathcal{R}^*$

of the signature. In summary, we have that:

$$\mathcal{R} = \{(a_i, A_i) | i \in \mathbb{N}^{[1,n]}\}$$

and for every $r \in \mathcal{R}^*$:

$$r = \{(a_i, v_i \in A_i) | i \in \mathbb{N}^{[1,n]}\}$$

Note that an entity does not have to instantiate every attribute of its entity type; the attributes without value assigment are simply omitted from the specification.

Uniqueness is an important property for entities in a database. It is often a hard requirement enforced by the database system that entities must be uniquely identifiable by some combination of their attribute values. If this were not the case, it would be possible to enter two different entities that become indistinguishable after being stored, representing an immediate loss of information and usability in general. For a given entity type, each irreducible subset of its attributes that can be used to uniquely identify entities from its body is called a *key*. In some cases, depending on which attributes are available (measured) and which are stored, it is possible that an entity type has no key. In such cases, to enforce uniqueness, an additional attribute (an artifical or surrogate key) is added to the entity type. An artificial key usually takes the form of an incremental integer number. Many database systems support this out-of-the-box. Using artificial keys is not without downsides: they introduce complexity (in the sense that a counter or some other value generator must be managed, either manually or by the database system) and metadata is explicitly injected and bound to each entity, strongly tying the entity to the particular database in which is it stored. If an entity is stored in multiple databases, its natural attributes are likely identical (or at least they should be) whereas its artificial key will be different. Moreover, the addition of surrogate keys do not have semantic value, and make the database harder to understand [1, 2, 25]. Nonetheless, a surrogate key also has significant benefits which some find so attractive that they introduce surrogate keys for all entity types, regardless of whether or not a natural key exists.

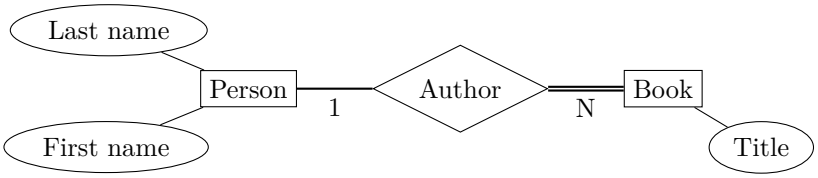


Figure 2.1: An example of an entity-relationship model, showing two entity types ("person" and "book") with a few attributes, and the relationship "author" between them. A book must have exactly one author, but a person can be the author of none or more books.

2.1.2 Relationship types

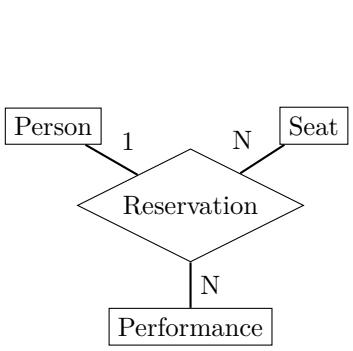
The second important component of ER modelling is a relationship. Similarly, ER-diagrams are actually display *relationship types*. Relationship types are used to express semantic links that would or should exist between entities of the connecting entity types. The relationships themselves exist only between particular entities. For example, there could be an entity type 'Person' and an entity type 'Book', with the relationship type 'Author' connecting them to establish the authorship of a person for a given book. We can categorize relationship types in several ways. A first way of doing this is by considering *cardinality restrictions*. Such a cardinality restriction states, for each entity type involved, whether either one or multiple entities can be involved in an instance of the relationship type in a body $\mathcal{R}^*$ of $\mathcal{R}$. For example, in the relationship type 'Author' as specified above, we want that a person can be the author of multiple books and a book can have multiple authors. As such, the cardinality restriction on both entity types is 'multiple'. If a relationship type involves precisely two entity types, we call this relationship type a binary relationship type. With respect to cardinality restrictions, there are exactly three types of binary relationship types: one-to-one, one-to-many and many-to-many.

A second way of categorizing relationship types, is by the *participation* of the involved entity types. More specifically, the participation degree of each involved entity type can be specified to be either mandatory (total participation) or optional (partial participation). If an entity type has total participation in a relationship type, then it is implied that each entity must participate in the corresponding relationship type. For example, if we want to enforce that each book must have an author, we can model this by stating that the entity type 'Book' has a total participation in the relationship type 'Author'. Conversely, when an entity type has partial participation in a relationship type, then it is implied that there can be entities that do not participate in the corresponding relationship type. In our example, we would want to model that not every person is the author of a book and thus model that entity type 'Person' has partial participation in the relationship type 'Author'.

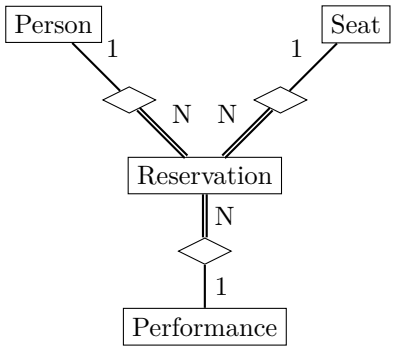
Though a relationship type can connect to more than two entity types, binary relationship types are the norm. In fact, more-than-binary relationship types can always be translated to a semantically equivalent entity type and a set of binary relationship types. This is demonstrated for a ternary relationship type (i.e., involving three entity types) in Figure 2.2

Relationship types can have attributes themselves, which can be useful to store properties that are related to neither of the entity types in particular but rather to their combination. For example, the price of a product depends not only on the product but also on the store.

Not every requirement can be modelled in ER. Things like domain restrictions, optional values and consistency constraints are examples of extra information that can not be enforced purely in an ER model. Such requirements are typically included in the form of a *functional description*.



(a) An implementation of an ER model for reservations using a ternary relation.



(b) An implementation of an ER model for reservations using only binary relations.

Figure 2.2

2.1.3 Data organisation

Data are stored on persistent media, such as hard disks, in locations referencable by an address. These *hard disk addresses* can be absolute addresses or logical addresses (in which case a translation table is used to map the logical addresses to absolute addresses) but, in any case, always point to a region (disk block) on the hard disk. These regions are an important concept because they form the basic unit of operation when interacting with hard disks: read and write operations happen on entire regions simultaneously. It is no trivial task to organise data efficiently, as deletions tend to cause *fragmentation* (i.e. there are "gaps" on the hard disk). Database systems are essentially just highly specialized software that excel at managing data. Nonetheless, database systems can only ensure that data are grouped if they are manipulated in a single operation. As such, the attribute values that describe a single entity are usually stored as a contiguous unit, as these data are manipulated in single operations (a new entity is inserted, deleted and updated as a whole rather than a sequence of separate attribute value insertions), but different entities are usually scattered across different hard disk addresses as they are manipulated separately and gradually over time. Database systems are software that specializes in managing (large) collections of addresses.

2.1.4 Querying

Database systems can be interacted with through a *query* protocol. After connecting to the database, a client can send a query to select, update or delete existing data, insert new data or to modify the configuration parameters of the database system itself.

Select queries are used to search for entities that satisfy a filter predicate. This filter predicate is used to describe preferred property values and the query

is evaluated in order to identify which (if any) entities have these preferred property values. Such a filter predicate can be simple (attribute a should take value v) but it can also be highly advanced, existing of a hierarchical composition of elementary criteria and logical connectives (prevalent in decision support systems). Regardless, a filter can be seen as a (Boolean) function that can be applied to an entity in order to test whether or not that entity should be returned as part of the *query result*.

In order to compute the result set of an arbitrary select query, a database system has to subject each entity to the test, a process that scales linearly with the amount of entities. Furthermore, in order to be able to apply the filter predicate on an entity, its data must be copied from the hard disk into memory, which is an expensive operation to perform. Nevertheless, the data from all entities that make up the result set must unavoidably be loaded. The challenge in optimizing query performance lies in avoiding as many evaluations as possible of entities that do not belong in the result set. This is especially interesting for select queries where one is searching for a small set of particular entities (perhaps just one entity), in which case linearly evaluating all entities would imply a vast majority of unnecessary load operations.

2.2 Fuzzy set theory

In 1965, Lotfi Zadeh introduced the concept of fuzzy sets as an extension to regular sets [26]. For a universe of discourse $\mathcal{U}$, a regular set $V \subseteq \mathcal{U}$ is characterized by a *membership function* $\mu_V : \mathcal{U} \to \{0, 1\}$, such that:

$$\forall u \in \mathcal{U} : \mu_V(u) = \begin{cases} 1 & u \in V \\ 0 & u \notin V \end{cases} \tag{2.1}$$

Hereby, 1 is interpreted as 'is element of the set' and 0 as 'is not element of the set'.

A fuzzy set $\widetilde{V}$ over $\mathcal{U}$ is characterized by an (extended) membership function $\mu_{\widetilde{V}} : \mathcal{U} \to [0, 1]$, which differs from a regular membership function in the sense that the discrete set $\{0, 1\}$ is extended to the unit interval $[0, 1]$. The value associated to an element $u \in \mathcal{U}$, $\mu_{\widetilde{V}}(u) \in [0, 1]$ is called the *membership grade* and is interpreted as follows:

1. 0 implies 'is not an element of the set'
2. 1 implies 'is fully element of the set'
3. $i \in]0, 1[$ implies 'is partially element of the set to degree i'

It is clear that a regular set is a special case of a fuzzy set where each element that is element of the set, is fully element of the set.

A fuzzy set $\widetilde{V}$ can be described by

$$\widetilde{V} = \{(x, \mu_{\widetilde{V}}(x)) \mid x \in \mathcal{U} \wedge \mu_{\widetilde{V}}(x) > 0\}$$

The elements with membership grade 0 are not included in the enumeration.

A fuzzy set $\widetilde{V}$ is further said to be *normalized*, if and only if it contains at least one element fully:

$$\exists u_1 \in \mathcal{U} : \mu_{\widetilde{V}}(u_1) = 1$$

2.2.1 Applicability for describing (uncertain) values

In the previous discussion on precision and digital data representations, we have already motivated the value of a representation with degrees of belief. Given the definition of fuzzy sets, it is clear that they are a natural fit for dealing with data imperfections.

In what follows, we will assume we are dealing with *singular attributes* (i.e., each attribute can only take one single *true value* at any given time). For brevity, we will just refer to these as 'attributes', which should be assumed singular unless otherwise stated. The goal is to find a way to translate any form of information regarding an attribute value to a mathematical model in such a way that as little information as possible is lost.

Applicability regarding uncertainty

When the information regarding an attribute value is uncertain in the sense that we can not say which value is the true value but we can rule out some values, we have incomplete information. In modal logic, each possible value corresponds to a *possible world* and is consistent with reality if the true world is one of the possible worlds [19]. Fuzzy sets can be used to represent possible worlds by mapping each possible value to a non-zero degree of possibility. Note that the interpretation of possible worlds is *disjunctive* (i.e. the true value corresponds to exactly one possible world, but not to multiple possible worlds simultaneously). In other words, the fuzzy set should be interpreted as a model for uncertainty regarding a single, true world.

Consider, for example, the statement "John is between 30 and 35 years old". If we're satisfied with year-precision, we can say

$$\mu_{\text{age}}(\text{John}) = \{(30, 1), (31, 1), (32, 1), (33, 1), (34, 1), (35, 1)\}$$

The correct interpretation of this fuzzy set is that the available information describes six, equally possible worlds and that John's true age is described by one of these. The membership grade that is associated to each world (here: 1) is not important, what matters is that each value is equally possible.

Applicability regarding imprecision

Fuzzy sets can be used to express imprecision. For example, when querying a real-estate data set, an example user preference regarding the price could be "not too expensive". Rather than specifying a hard boundary cut-off, one could construct a fuzzy set to represent the desired relation that more expensive implies less preferrable. It is tricky to specify a (normalized) fuzzy set when no

information is given regarding the maximal real-estate price but this need not be a problem, because one can still use a hard boundary cut-off as a threshold beyond which real-estate no longer needs to be ranked because they can all be rejected as being "too expensive". The added value of fuzzy sets lies in the fact that all real-estate with a price below this threshold will be sorted according to the preference function rather than just returned in an unordered result set.

(image of example fuzzy set)

Note how in this example, the fuzzy set does not express uncertainty. Here, the membership degrees should be interpreted as indicators of how strongly each value is preferrable (also: degrees of *truth*[28], *satisfaction*, *preference*[6] or *suitability*)[11]. The fuzzy set is said to be *conjunctive* as it is a construct for describing a whole collection of values and an order relation over all of them.

It should already be mentioned here that fuzzy sets are subjective in the sense that different users might have a different interpretation of the same concept. This is intentional and the flexibility of fuzzy sets is regarded as one of its strengths. However, it also implies that the results of an evaluation using fuzzy sets to express vagueness are most meaningful to the user who constructed the query.

2.2.2 Possibility Theory

The reader should be aware of the significant difference between using fuzzy sets to express "too expensive" when formulating a preference on the one hand and using a fuzzy set to describe the price of a particular real-estate as "too expensive" on the other hand: the former is conjunctive and certain (a precise model for vagueness), the latter disjunctive and uncertain. Fuzzy sets that denote uncertainty are examples of *possibility distributions*. It was first introduced by Lotfi Zadeh in [27] as an application of his theory of fuzzy sets. The theory of possibilities was further pioneered by Didier Dubois, Henri Prade and Gert De Cooman [9, 8, 12, 13, 14].

Central to possibility theory are *possibility measures*. A possibility measure Π is a formal model for describing uncertainty regarding the true value of an attribute with domain $\mathcal{U}$. Let $\mathcal{P}(\mathcal{U})$ be the powerset of $\mathcal{U}$ and $\mathcal{B}$ a belief space. Let further $\inf \mathcal{B}$ denote the minimal degree of belief, signifying 'impossibility', and $\sup \mathcal{B}$ the maximal degree of belief, signifying 'certainty'. Any possible world-based uncertainty model $\mathrm{G} : \mathcal{P}(\mathcal{U}) \to \mathcal{B}$ is essentially a *fuzzy measure* [16, 24] and must adhere to three axioms:

1. $\mathrm{G}(\emptyset) = \inf \mathcal{B}$
2. $\mathrm{G}(\mathcal{U}) = \sup \mathcal{B}$
3. $\forall U_1, U_2 \subseteq \mathcal{U}, U_1 \subseteq U_2 : \mathrm{G}(U_1) \leq \mathrm{G}(U_2)$

The first axiom states that a true value must exist (it is impossible that the attribute takes no value). The second axiom implies that the domain is complete and that it contains the true value (the attribute must take some value from the

domain). The third axiom entails monotonicity, meaning that adding possible values to a set can not result in a decrease in belief.

A possibility measure satisfies these axioms by implementing the monotonicity constraint by means of maxitivity in the belief space $[0, 1]$:

1. $\Pi(\emptyset) = 0$
2. $\Pi(\mathcal{U}) = 1$
3. $\forall U_1, U_2 \subseteq \mathcal{U} : \Pi(U_1 \cup U_2) = \max(\Pi(U_1), \Pi(U_2))$

For some event U, $\Pi(U)$ is called the *possibility* that event U occurs. Possibility measures are special in the sense that they are the largest measures satisfying the three axioms stated above. This means they are the least restrictive in combining uncertainty of two subevents. Conversely, we can also consider the smallest measures satisfying the axioms. These measures are called *necessity* measures and denoted by N. For necessity measures, monotonicity is obtained by minitivity:

1. $\mathrm{N}(\emptyset) = 0$
2. $\mathrm{N}(\mathcal{U}) = 1$
3. $\forall U_1, U_2 \subseteq \mathcal{U} : \mathrm{N}(U_1 \cap U_2) = \min(\mathrm{N}(U_1), \mathrm{N}(U_2))$

Possibility and necessity measures are closely related in several ways. First, we have that:

$$\mathrm{N}(U) = 1 - \Pi(\bar{U}) \tag{2.2}$$

where $\bar{U}$ denotes the opposite event of U, i.e. $\mathcal{U} \setminus U$. Moreover, from axioms 2 and 3 of both measures, it follows that either $\Pi(U) = 1$ or $\Pi(\bar{U}) = 1$. In other words, for any U, if $\Pi(U) < 1$, then $\mathrm{N}(U) = 0$. It follows that $\mathrm{N}(U) \leq \Pi(U)$.

Combined, necessity and possibility form a total order relation on the powerset of the universe of discourse. This order introduces a clear interpretation of necessity and possibility, and is specified as follows:

- if $\mathrm{N}(U) = 1$, U is certainly true.
- if $\Pi(U) = 1$, U is possible - it would not be surprising if U were true.
- if $\mathrm{N}(U) = 0$, U is unnecessary - it would be surprising if U were true.
- if $\Pi(U) = 0$, U is certainly not true.

The distance between the possibility and necessity of an event U (i.e.: $\Pi(U) - \mathrm{N}(U)$) conveys information regarding the certainty about whether or not U is true. If there is no information about U, this distance should be maximal, so we have that $\mathrm{N}(U) = 0 \ \wedge \ \Pi(U) = 1$.

For universes that are finite or countably infinite, a possibility measure can be expressed in terms of its behavior on singletons:

$$\Pi(U) = \max_{u \in \mathcal{U}} \Pi(\{u\}) \tag{2.3}$$

The collection of possibilities for all singletons is called the *possibility distribution* Π:

$$\pi(u) \triangleq \Pi(\{u\}), \forall u \in \mathcal{U} \tag{2.4}$$

Through axiom 3 it is straightforward to reconstruct an entire possibility measure from a possibility distribution. From axioms 2 and 3, it follows that there must be at least one single value in $\mathcal{U}$ that is fully possible:

$$\exists u_1 \in \mathcal{U} : \pi(u_1) = \Pi(\{u_1\}) = 1 \tag{2.5}$$

Note that this immediately implies that the necessity of all values other than u_1 must equal 0, because:

$$\begin{aligned} \forall u \in \mathcal{U} \setminus \{u_1\} : \mathrm{N}(\{u\}) &= 1 - \Pi(\mathcal{U} \setminus \{u\}) && (2.6)\\ &= 1 - \max_{x \in \mathcal{U} \setminus \{u\}} \Pi(x) && (2.7)\\ &= 1 - \Pi(\{u_1\}) && (2.8)\\ &= 0 && (2.9) \end{aligned}$$

In other words, only u_1 can have a non-zero necessity. Furthermore, u_1 only has a non-zero necessity if and only if there exists no other value $u_2 \in \mathcal{U}, u_2 \neq u_1$ that is also fully possible. Namely, if $u_1 \neq u_2$ and $\Pi(\{u_2\}) = 1$, then $u_2 \in \mathcal{U} \setminus \{u_1\}$ and thus $\Pi(\mathcal{U} \setminus \{u_1\}) = 1$ which implies $\mathrm{N}(\{u_2\}) = 0$ (and vice versa for u_1).

Clearly, a projection of the necessity measure over the singletons of a universe of discourse would not be very useful, as either only one value has a non-zero necessity, or all values are unnecessary. Moreover, because necessity is defined in terms of intersections, no further information can be derived from the necessities of singletons. As such, unlike a possibility distribution, a "necessity distribution" could not be used to reconstruct the necessity (or possibility) measure.

One could, however, construct a (mathematically) informative distribution based on the necessity measure by associating each singleton u to the necessity of $\mathcal{U} \setminus \{u\}$. This could be considered useful because it contains sufficient information to reconstruct the entire necessity measure (and consequently also the possibility measure) using intersections. However, from the relation between necessity and possibility it is easy to show that this distribution is essentially equal to the 1-complement of the possibility distribution, which, despite capturing the same amount of information, is harder to interpret than the possibility distribution itself, without adding extra information.

For infinite universes (continuous quantities), the possibility measure is typically approximated by a *shape function*. For such quantities, this implies that the domain can be represented by a one-dimensional axis.

Perhaps the most frequently used shape function is the trapezoidal shape function Π, characterized by a four-tuple of domain values (a, b, c, d) such that $a \leq b \leq c \leq d$, which is defined as follows:

$$\Pi(.; a, b, c, d) : \mathcal{U} \rightarrow [0, 1] \tag{2.10}$$

$$u \rightarrow \begin{cases} 0 & \text{if} u < a \\ \frac{u-a}{b-a} & \text{if} a \leq u < b \\ 1 & \text{if} b \leq u \leq c \\ \frac{d-u}{d-c} & \text{if} c < u \leq d \\ 0 & \text{if} d < u \end{cases} \tag{2.11}$$

Plotting this function (Figure 2.3) quickly reveals why it is called the trapezoidal shape function. When $b = c$, the shape function becomes triangular and when $a = b \wedge b < c \wedge c = d$, it becomes rectangular. The case wherein $a = b = c = d$ incidates there is no uncertainty. It is easy to see that changing the specificity of the domain has an influence on the difference between b and c, and how a trapezoidal shape function can be collapsed to a triangular one by zooming out and vice versa by zooming in.

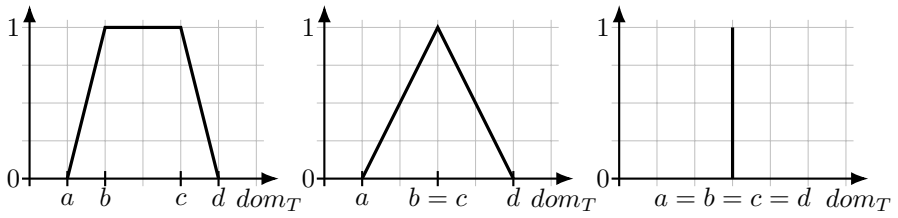


Figure 2.3: Examples of trapezoidal possibility distributions. From left to right: an uncertain value that is most likely between b and c, an uncertain value that is most likely b $(= c)$, and a certain value.

Similarities to Probability Theory

Probability theory [4, 22] is the most established mathematical framework for representing uncertainty as a belief model. In this framework, the belief measure is called the probability measure Pr and degrees of belief are called probabilities. Probability theory was mathematically formalized by Kolmogorov [18] in a set of axioms:

1. $\forall U \subseteq \mathcal{U} : \Pr(U) \geq 0$
2. $\Pr(\mathcal{U}) = 1$
3. $\Pr\left(\bigcup_{i=1}^{\infty} E_i\right) = \sum_{i=1}^{\infty} \Pr(E_i)$

It is easy to show that these satisfy the three axioms of a generic belief model stipulated above, where this time σ-*additivity* implements the monotonicity axiom.

Similar to a possibility distribution, probability theory has the concept of a *probability mass function* (pmf) which shows how probability is distributed across a finite (or countably infinite) amount of possible worlds. For a continuous quantity, an alternative distribution called a *probability density function* (pdf) must be used because it is impossible to select the exact world that corresponds to the real world due to the precision problem. As such, $pdf(u_1)$ represents the density of world u_1, and the probability of an approximate value (i.e. a region U_1 in the domain) can be found by integrating this curve:

$$\int_{U_1} pdf(u_1)\mathrm{d}U_1$$

Interpretations of Probability

Given probability theory, which is well understood and widely accepted, why do we need possibility theory? For starters, it is no secret that there are different interpretations to what exactly probability degrees represent. There are two broad categories: objective, frequency-based, physical probability and subjective, Bayesian, evidential probability.

One of the historically earliest interpretations relies on the assumption that possible outcomes can be grouped in classes that are equally likely. Examples are coin flips and tosses with a fair die. Under these assumptions, the probability of a specific set of outcomes is exactly defined as the ratio of the amount of favorable outcomes over the total amount of outcomes.

According to frequentists, probability is an indicator of how frequently a possible world was observed after repeating a stochastic experiment a large amount of times. There is no presumption that each outcome is equally likely up front, the likelihood of each outcome is determined through observation. If the experiment would be repeated an infinite amount of times (which is impossible in practice), the observed frequency of each possible world would converge to its probability of occurring the next time the experiment is carried out. Criticism on this approach includes that it can not be applied in situations where the experiment can not be repeated, and even if it could, it is debatable whether or not it is actually possible to repeat the experiment under identical circumstances and how many times it should be repeated in order for the conclusions to be representative.

In the Bayesian interpretation (based on Bayes' theory of Bayesian inference but actually pioneered by Laplace), probability is a subjective indicator of belief. Different agents might assign different probabilities to different outcomes and there is no reason for someone to believe one agent over the other. Bayesian probabilities further also rely on the assumption that agents are perfectly rational [3, 21], though evidence has shown that this claim is doubtful [17]. This interpretation is often applied in the context of betting games, to set payout

rates for outcomes in correspondence to their perceived probability of occurring, which is immediate evidence that perceived probabilities can be used to turn a profit and are not necessarily fair. Au contraire, this interpretation relies on the idea that the true likeliness of the outcomes can be hidden behind slightly manipulated probabilities in order to trick gamblers into entering an unfair game, the lottery being a prime example thereof.

Differences with Probability Theory

While probability theory is ideal for dealing with stochasticity, it can lead to the fabrication of false information when applied to epistemic uncertainty stemming from a lack of information. There is in fact a large difference between the outcome of the roll of a die and the incomplete knowledge regarding the true color of John's car given that it is only known to be a "dark" color. For the former, the outcome is inherently random though the experiment itself is perfectly known. For the latter, the experiment of measuring the color of the car is not stochastic and the uncertainty stems purely from the fact the color of the car simply hasn't been accurately measured yet. The following example serves to illustrate how applying probability theory in a setting of incomplete information can lead to false knowledge.

Example. Assume there is a wooden cube of which nothing is known except that the length of its edges is between 3 and 5 centimeters. According to the principle of symmetry (or the principle of indifference, named by Boole but introduced earlier by Bernoulli joined by Laplace, who deemed it to be intuitively obvious), one should model the lack of information by a uniform distribution, in order to reflect that each possible outcome is considered equally believable. This concept also appears in Bayesian terms in the form of the least informative prior. For frequentists, this step is already meaningless because there are no previous iterations to obtain these relative frequencies from.

A similar reasoning can be applied to the knowledge regarding the volume of the cube, besides that it must be somewhere in between 3^3 and 5^3. Consistently, our lack of knowledge should here also yield a uniform distribution over all possible volumes in this range. However, due to the physical relation between the edge of a cube and its volume, it can easily be calculated that if the length of the edge is assumed to be uniformly distributed, the volume can not be uniformly distributed; it would be biased towards lower volumes. This should lead us to believe that there is some form of information, of evidence, that suggests that it is more likely that the volume of the cube is closer to 3^3 than to 5^3, yet this contradicts the prior assumption that nothing is known about the cube's volume. Dually, assuming a uniform distribution over the possible volumes would imply evidence that the length of the sides of the cube is closer to 5 than to 3.

The devil lies in the additivity, which makes it impossible to simultaneously impose a uniform distribution on two related attributes: if the length of the edge

of the cube is uniformly distributed, it is implied that its volume is not, and vice versa. Additivity and dependencies between variables appear frequently throughout probability theory and always introduce cases that must be treated separately, often also mathematically more complex. It is not atypical for novice statisticians and scholars to work under the safe umbrella where it is assumed that variables are considered independent. Obviously, this is not always the case in reality. Regardless, the example above shows why probability theory can and should not be used to represent a simultaneous lack of information regarding two or more related properties of an entity.

The scenario is different for possibility theory. The principle of indifference can be applied for here as well and states that a possibilistic uniform distribution should be used, assigning all possible values to the same possibility degree. Because of the use of maxitivity, the possibilistic uniform distribution can assign a possibility degree of 1 to each world rather than some normalized probability or density degree. An additional advantage is that the possibilistic approach is identical for both the discrete and continuous cases, whereas in probability theory one would have to differentiate between a mass function and a density function, both of which have to be handled differently in order to obtain information from them. Because of the maxitivity implementation of the monotonicity axiom, it is possible to specify a possibilistic uniform distribution for both the edge and the volume of the cube. Moreover, it can easily be verified that starting from a possibilistic uniform distribution for either and applying the physical relation to deduce the distribution for the other again results in the possibilistic uniform distribution, proving that it is not only possible but even implied by possibility theory that starting without information can not lead to the (false) deduction of knowledge.

There are plenty of examples of paradoxes based on these observations, like the rising sun paradox, the doomsday paradox, the Bertrand paradox, and so on. This does not mean that we should stop using probability theory but simply implies that it is not correct to use additivity when using a subjective model for uncertainty.

Construction of Possibility Distributions

One thing probability theory does have that possibility theory lacks, is a strong foundation for specifying how possibility degrees for events can be calculated. Regardless of the interpretation you prefer, there is plenty of material to help you compute the probability measure of an experiment (attribute) over its outcomes (domain). It is not so clear, however, how possibility distributions should be constructed. It can be expected, though, that when modelling a term that implies a monotonic relation with the natural order over the domain (e.g. “old age”), the possibility distribution is also monotonic [12, 15, 23].

In such circumstances, a possibility distribution is no longer necessarily subjective but should be interpreted as a total order relation over its domain. For infinite domains, however, specifying a total order relation implies $a = -\infty$ and $b = +\infty$, which is not practical towards computations. Indeed, in doing so it

becomes impossible to derive the precise possibility degree for a particular domain value, and only comparisons remain. In order to avoid this, the possibility distribution is instead usually defined with a clear minimum and maximum. When specified with care, these bounds do not need to limit the usability of the possibility distribution. In fact, good bounds can often be derived naturally from the context in which the modelled term is used. When using "old" to describe people's age in years, for example, acceptable bounds could be 0 for the minimum and 120 for the maximum, assuming that people do not typically become 120 years old.

Non-monotonic terms (e.g.: "middle-aged") do not align with a certain order, but rather with a particular range of values. Such terms are best expressed using fuzzy intervals: fuzzy sets whose membership function is trapezoidal and for which $a \neq b \wedge c \neq d$. The question is then which values to choose to define the fuzzy interval to describe "middle-aged". This is clearly subjective: what one person considers "middle-aged", someone else might find "young".

In any case, the possibility degree of some value is most informative when it is not interpreted individually but rather in comparison to the possibility degrees of other values. In other words, possibility distributions are best treated as a whole, which supports our fundamental conviction that attribute values should be denoted with fuzzy sets rather than single values. For a more thorough discussion, we would like to refer to [10].

2.3 Aggregation of individual criteria

We have already briefly introduced how fuzzy sets can be applied to model imprecision, for example in querying, and that this application of fuzzy sets is in no way related to uncertainty.

Recall the real-estate example where we specified a user preference regarding price as "not too expensive". Intuitively, this would correspond to a monotonically decreasing relation between price and its corresponding preference. Evaluating the query essentially comes down to *sorting* the data according to the (weak) order that is imposed by the fuzzy set representing the preference.

It might seem unnecessary to work with fuzzy sets when all it accomplishes is a simple sort operation, but the benefits of encorporating fuzzy sets in query evaluation become more apparent as soon as more than one attribute is taken into account in the evaluation process. Indeed, as soon as there are as little as two attributes under consideration, sorting is no longer trivial. Consider for example the products catalogue of a retailer. There are many different ways to sort products: by price, by release date, by amount in stock, by vendor, by model, by review score... Though it is possible to sort the products by a preference on any of these properties individually (increasing price, good review score, recent release), the order will almost always be different for each. And what if we want to sort things by a combination of these properties, rather than by only one at a time (e.g. low price *and* good review score)?

There are different strategies for forcing a sort order on a multidimensional

data. One of them is to choose an attribute priority and to sort according to the natural order of the attribute with highest priority first, only relying on the next attribute in the priority list to distinguish between entities with an identical value for the first attribute, and so on. This is called lexicograhical order, and despite its simplicity and advantage that the resulting ranking is very straightforward to understand, it only has limited flexibility in the possible rankings that it can produce. It also requires some discretization when it is applied to continuous quantities in order to determine when two attribute values can be considered equal, because otherwise no further distinctions could be made. Applying this to the real-estate example, one could consider ascending price as the first priority and distance to work as the second. Perhaps it can be said that price is more important than distance, though it is harder to justify that one would not be willing to pay as little as 1 cent more for a house that is 100 kilometers closer to work. Clearly, there is a need for a more nuanced approach.

A broad spectrum of alternatives can be found in the category of (weighted) aggregation techniques, where different attributes are considered simultaneously instead of independently. For example, the price may be assigned an importance of 60 and the distance an importance of 40 to indicate price is 1.5 times more important than distance, meaning that a change of 1 price unit is canceled out by an opposite change of 1.5 units in distance, nullifying the previous situation wherein the lexicographical order makes no balance between price and distance, which led to an unintuitive result. One other effect of using aggregation techniques, however, is that the overall sorting is no longer trivial to reverse engineer, as different entities might receive a similar (mediocre) score despite having very different attribute values.

Adding weights to attributes makes it possible to express relative importance between attributes, but only regarding (strictly) monotonic preferences over their domains (lower price, smaller distance). On top of this limitation, there is also no real frame of reference regarding *how good* the best solution is: is some solution the best solution because it strongly satisfies the preferences or because it is simply better than all other alternatives?

Fuzzy sets alleviate these problems by forcing users to specify preference mappings that effectively translate attribute values from different domains to the same preference space, making it easy to compare them. Working with standard, normalized fuzzy sets, one would specify preferences such that values are associated to a score of 0 in case they are undesirable and 1 in case they are ideal. Though it is common to do so, it is not strictly necessary to assign the preference space to the unit interval. Alternatively, one might choose a bipolar approach, specifying undesirable values by associating them to degrees < 0 and preferrable values to degrees > 0, reserving 0 for values that the user is indifferent about [7, 29, 20, 5]. However, it can easily be shown that such spaces are bijective to the unit interval. As we will come back to later, the default choice for the unit interval has proven to be a dangerous one, as it has been the reason for quite some nonsensical research results due to applying calculus to values with completely different semantics, a mathematical trick that was made

possible by there being different quantities that are all expressed in the unit interval.

By introducing fuzzy sets in the query evaluation process, it is immediately clear that a result with a score of 1 exemplifies the ideal solution. Not only are the results comparable, but the overal score also has an absolute meaning.

Commercial platforms typically offer some tailored formula for creating a custom sorting[1], but in industrial applications, it makes sense that data analysts have full control over which evaluation function is used to create an overall ranking that takes all dimensions into account and that can be well explained to decision makers. Though small problems may be tackled heads-on, large problems can have tens or even hundreds of properties to be considered, clearly underlining the usefulness of automating the evaluation process to be done by a computer. This research area is called *decision support*. Not only do computers help when problems grow in size, but also when the set of alternatives to compare becomes large. Deciding between a handful of options (which movie to see with friends at the theatre) can be done manually but evaluating millions of alternatives in an objective manner in order to obtain a global ranking without computerized aid is challenging to say the least.

However, in order to be able to employ computers to do the heavy lifting for us, we need to find a way to encode our (fuzzy) evaluation logic in an algorithm. Under the assumption that, for a particular user, a specific (weak) order among a set of systems reflects how well these systems satisfy their preferences, the challenge is thus to provide tools that allow that user to express their preferences in the form of an evaluation function in such a way that applying that function sorts the systems according to the aforementioned (weak) order.

Such frameworks are designed with two large goals: the intuitiveness when translating human evaluation logic on the one hand and the degree to which the results are (not) surprising to the user modelling their preferences.

The intuitiveness of an approach is largely perceived as the notion of how well inter-criteria relations can be specified. In logic, the term *quantifier* is used to denote a relation between criteria. Examples of commonly known quantifiers are the existential and universal ones, denoting that respectively any and all criteria must be satisfied, which can be translated to the logical connectives 'or' and 'and'.

For practical problems, however, it is rarely the case that we strictly want that exactly any or all criteria are met. Some examples of other quantifiers are 'at least X', 'most' and 'about X'. An example of a more advanced connective is 'X is mandatory, Y is desired and Z is optional'.

There are several aggregation frameworks that incorporate fuzzy set theory in order to support such connectives. Despite their differences, they all use some form of weighted average to aggregate the preference degrees that correpond to elementary criteria. In this thesis, we will investigate what changes when values are allowed to be uncertain.

[1] It is well known that Google uses a highly advanced algorithm to personally tailor search results to each of its users.

2.4 Summary

From these preliminaries, the fundamentals of how data is stored on and retrieved from digital system should be clear. Data are typically structured as entities with relationships between them. In order to be able to retrieve specific entities rapidly, indices (which essentially are just storage-optimized copies of subsets of the entire data set) are used.

In order to make computers capable of dealing the fuzzy way people naturally employ to decribe and process information, a mathematical framework for translating vagueness into a precise, numerical domain is necessary. Fuzzy set theory describes an essential way of doing so. Not only can fuzzy sets be used to represent states of perfect, incomplete or total lack of knowledge regarding data, it is also useful for flexible querying. Flexible querying allows people to also express preferences using concepts from natural language, which are, unlike classical predicates, fuzzy by nature. Not only does this make it trivial to rank query results from best to worst, but it avoids the possibility that an almost perfect entity is rejected because it falls just outside of some arbitrarily defined threshold, an effect that is most painfully noticeable when entities are rejected due to a miniscule measuring error. Furthermore, flexible querying opens the door to flexible aggregation, a technique to combine the evaluation results of multiple criteria.

The largest remaining issues are how to translate indexing and flexible querying to uncertain data. These topics will be covered in the next chapters.

Bibliography

[1] Chris J Date. *An introduction to database systems*. Vol. 1. Pearson Education India, 1977.

[2] Chris J Date. *Relational Database Writings, 1994-1997*. Vol. 4. Addison-Wesley Longman, 1998.

[3] Bruno De Finetti. "Foresight: Its logical laws, its subjective sources (1937)". In: *Studies in subjective probability* (1980), pp. 55–118.

[4] Bruno De Finetti. *Theory of probability: A critical introductory treatment*. Vol. 6. John Wiley & Sons, 2017.

[5] Guy De Tré. "Extended possibilistic truth values". In: *International Journal of Intelligent Systems* 17.4 (2002), pp. 427–446.

[6] Didier J Dubois. *Fuzzy sets and systems: theory and applications*. Vol. 144. Academic press, 1980.

[7] Didier Dubois and Henri Prade. "Bipolarity in flexible querying". In: *Flexible query answering systems* (2002), pp. 174–182.

[8] Didier Dubois and Henri Prade. *Possibility theory: an approach to computerized processing of uncertainty*. Springer Science & Business Media, 2012.

[9] Didier Dubois and Henri Prade. "Possibility theory: qualitative and quantitative aspects". In: *Quantified representation of uncertainty and imprecision*. Springer, 1998, pp. 169–226.

[10] Didier Dubois and Henri Prade. "Practical Methods for Constructing Possibility Distributions". In: *International Journal of Intelligent Systems* 31.2 (2015), pp. 215–239. ISSN: 08848173. DOI: 10.1002/int.

[11] Didier Dubois and Henri Prade. "The three semantics of fuzzy sets". In: *Fuzzy sets and systems* 90.2 (1997), pp. 141–150.

[12] Gert De Cooman. "Possibility theory 1: The measure- and integral-theoretic groundwork". In: *International Journal of General Systems* 25.4 (1997), pp. 291–323.

[13] Gert De Cooman. "Possibility theory 2: Conditional possibility". In: *International Journal of General Systems* 25.4 (1997), pp. 325–351.

[14] Gert De Cooman. "Possibility theory 3: Possibilistic independence". In: *International Journal of General Systems* 25.4 (1997), pp. 353–371.

[15] Gert De Cooman. *Towards a possibilistic logic.* 1995.

[16] Michel Grabisch. "K-order additive discrete fuzzy measures and their representation". In: *Fuzzy sets and systems* 92.2 (1997), pp. 167–189.

[17] Daniel Kahneman and Patrick Egan. *Thinking, fast and slow.* Vol. 1. Farrar, Straus and Giroux New York, 2011.

[18] Andreĭ Nikolaevich Kolmogorov and Albert T Bharucha-Reid. *Foundations of the theory of probability: Second English Edition.* Courier Dover Publications, 2018.

[19] Saul A Kripke. "A completeness theorem in modal logic". In: *The journal of symbolic logic* 24.1 (1959), pp. 1–14.

[20] Tom Matthé and Guy De Tré. "Bipolar query satisfaction using satisfaction and dissatisfaction degrees: bipolar satisfaction degrees". In: *Proceedings of the 2009 ACM symposium on Applied Computing.* ACM. 2009, pp. 1699–1703.

[21] Frank P Ramsey. "Truth and probability". In: *Readings in Formal Epistemology.* Springer, 2016, pp. 21–45.

[22] Cedric AB Smith. "Consistency in statistical inference and decision". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 23.1 (1961), pp. 1–25.

[23] Peter Walley and Gert De Cooman. "A behavioral model for linguistic uncertainty". In: *Information Sciences* 134.1-4 (2001), pp. 1–37.

[24] Zhenyuan Wang and George J Klir. *Fuzzy measure theory.* Springer Science & Business Media, 2013.

[25] R Wueringa and W de Jonge. *The identification of objects and roles-Object identifiers revisited.* Tech. rep. Technical Report IR-267, Faculty of Mathematics and Computer Science, 1991.

[26] L.a. Zadeh. "Fuzzy sets". In: *Information and Control* 8.3 (1965), pp. 338–353. ISSN: 00199958. DOI: 10.1016/S0019-9958(65)90241-X.

[27] Lotfi Zadeh. "Fuzzy Sets as a Basis for Possibility". In: *Fuzzy Sets and Systems* 1 (1978), pp. 3–28.

[28] Lotfi A Zadeh. "The concept of a linguistic variable and its application to approximate reasoning". In: *Information sciences* 8.3 (1975), pp. 199–249.

[29] Slawomir Zadrożny and Janusz Kacprzyk. "Bipolar queries and queries with preferences". In: *Database and Expert Systems Applications, 2006. DEXA'06. 17th International Workshop on.* IEEE. 2006, pp. 415–419.

Chapter 3

Querying Uncertain Data

When evaluating a query that consists of one or more criteria on a database, the result consists of all entities that satify the specified criteria. Regular query evaluation relies on two presumptions. On the one hand it is assumed that the data are perfectly known and accurately represent reality. On the other hand it is also assumed that the user does not only know exactly what they are looking for, but also that they are capable of describing it perfectly. Both presumptions are often not true in practice.

Databases capable of handling imperfect information attempt to alleviate the former presumption. As a result, either the user has to specify their stance on uncertainty such that the result set can remain a regular set, or any uncertainty regarding attribute values must be propagated to the result set (requiring that it be replaced by a representation of uncertainty, e.g. a fuzzy set).

Specifying a stance on uncertainty in the traditional sense implies choosing for a particular set of operators that extend classical operators so they can be applied to uncertain values. For instance, one can extend the classical operators such that their application on uncertain values (e.g. is "about 40" larger than 40?) still evaluates to a Boolean value. Naturally, each operator necessarily has multiple extensions that each reflect a different degree of tolerance towards uncertainty. For the *larger than* operator ($>$), for example, $>_1$ could denote "certainly larger than" and $>_2$ "possibly larger than", providing two alternative ways to generalize the operator $>$. Obviously, the result set of a query depends on which generalization is used: for a person whose age is described by "around 40", the evaluation of the query $>_1 40$ would be FALSE, whereas the evaluation of the query $>_2 40$ would be TRUE. In other words, this approach requires that the user submitting the query must rephrase their question in a way that is resilient to uncertainty.

Alternatively, in order to exempt the user from having to actively choose a strategy, a more-than-binary evaluation logic that is able to cope with uncertainty about truth can be employed, such that the uncertainty regarding cases where there is insufficient information to determine the truth-value of a given statement is reflected in the result. By replacing the implementation of query

criteria as predicates by functions that map to a three-valued space ({TRUE, FALSE and MAYBE}), for example, the result of the query "around $40 > 40$" can be evaluated to a value that reflects uncertainty (in this case MAYBE). To minimize information loss, cases where there are different degrees of belief regarding certain attribute values require a many-valued logic that maps each entity to a degree of confidence that it should be part of the result set. Generally, the result set of a query on uncertain data can be decribed by a fuzzy set where the membership degrees of the entities denote indicators of confidence, belief or possibility.

The second presumption, that users are capable of expressing their preferences with perfect accuracy, can be tackled by allowing users to be imprecise in their criteria. Any imprecision in the query criteria is reflected in the results by mapping each entity to a degree of suitability. As such, the result set can be seen as a fuzzy set where the membership degrees are interpreted as degrees of compatibility. An immediate advantage of using flexible criteria is that the entities in the database are ranked (rather than filtered) by their satisfaction degree, and that near-perfect solutions (that would typically be rejected) are not omitted from the result set and can still be consulted. For example, for a set of bottles, one could evaluate the flexible criterion expressing a desire for *full bottles*. Evaluating the flexible query would associate each bottle to a numerical degree in the unit interval indicative of the extent to which it is full (i.e. a full bottle will have degree 1, a half-empty bottle would lead to satisfaction degree 0.5, an empty bottle would have degree 0 and so on). Flexible criteria have been widely employed [29, 3, 25, 30, 20, 5] and are an essential basic component of many systems (including multi-criteria decision support [21]) for their ranking properties and their intuitive nature.

Ideally, both approaches can be combined so that both presumptions can be foregone and we can evaluate flexible criteria on uncertain data. This is challenging, however, as both solutions transform the result set into a fuzzy set, but each with a different interpretation. For example, if the content of a bottle is unknown and the answer to the query for "full bottles" returns 0.5, does that mean that the bottle is half full or that it is possible to degree 0.5 that it is completely full?

In this chapter, we will look at ways to evaluate flexible criteria on uncertain data, more precisely on how the results of such evaluations can be represented in a meaningful and useful way. First, we will give an overview of existing approaches in section 3.1. Then, we will propose a new approach in section 3.2, based on possibility theory. Its advantages, properties and semantics are discussed in sections 3.3 and 3.4. Afterwards, it is discussed how the concepts of the approach can be extended beyond possibility theory in section 3.5. Essential parts of this chapter have been published in [8].

3.1 Evaluating flexible criteria on uncertain data

Some fundamental contributions on the topic of evaluating flexible criteria on uncertain data include [4, 23, 30, 28, 10]. Dubois and Prade mention in [10] that "the complete answer to a query evaluating to what extent an agent believes a fuzzy proposition" (or else: to what extent an uncertain property satisfies a flexible criterion) is described by "a possibility distribution over truth values" (or thus a distribution of uncertainty over degrees of suitability). However, this is not further researched under the assumption such distributions are "possibly hard-to-interpret". They instead suggest the concept of extended possibility and necessity degrees. A similar idea can be found in an early study by Cayrol et al. regarding standard fuzzy pattern matching indices for fuzzy information systems [6]. Prade et al. [23] represent the evaluation of a flexible criterion c (modeled as a fuzzy set characterized by μ_c) on an uncertain property a (with domain A and represented by possibility distribution π) using a possibility Π and necessity N given by:

$$\Pi(a \text{ IS c}) = \sup_{v \in A}(\min(\pi(v), \mu_c(v))) \tag{3.1}$$

$$\mathrm{N}(a \text{ IS c}) = \inf_{v \in A}(\max(1 - \pi(v), \mu_c(v))) \tag{3.2}$$

These numbers semantically express respectively the possibility and the necessity that a is compatible with c (denoted by the IS operator). One should be cautious when interpreting these degrees because, as mentioned by Prade et al., some of the properties for possibility and necessity measures no longer hold. More precisely (let C denote the predicate a IS c), the following are no longer true:

$$\max(\Pi(C), \Pi(\bar{C})) = 1 \tag{3.3}$$

$$\min(\mathrm{N}(C), \mathrm{N}(\bar{C})) = 0 \tag{3.4}$$

An important consequence thereof is that the evaluation of a criterion can lead to possibility and necessity degrees that are simultaneously strictly between 0 and 1, e.g. $\Pi = 0.7$, $\mathrm{N} = 0.4$. This implies "it is not fully possible c is compatible, but it is also not fully possible c is not compatible" [10].

The interpretation of the degrees Π and N generated through these formulas is ambiguous: they are not strictly indicators of uncertainty nor are they bounds for a degree of satisfaction, though they seem to have properties of both. It can be argued that these muddled semantics are the consequence of combining and comparing degrees of satisfaction with degrees of possibility during their computation. The mathematics only seem sensible because it was chosen to express both degrees of satisfaction and possibility in the unit interval, though it should be clear that such degrees should not be compared. Indeed, using different numerical scales, which is perfectly allowed, would make the proposed formulas nonsensical. Imagine for example that degrees of satisfaction are expressed over the range $[-1, 1]$ and degrees of possibility over the range $[0, 100]$. Comparing degrees of satisfaction and degrees of possibility is the figurative equivalent

of comparing apples and oranges, and should not be done under any circumstances. This conclusion is in fact discussed at length and supported by Dubois and Prade in [10].

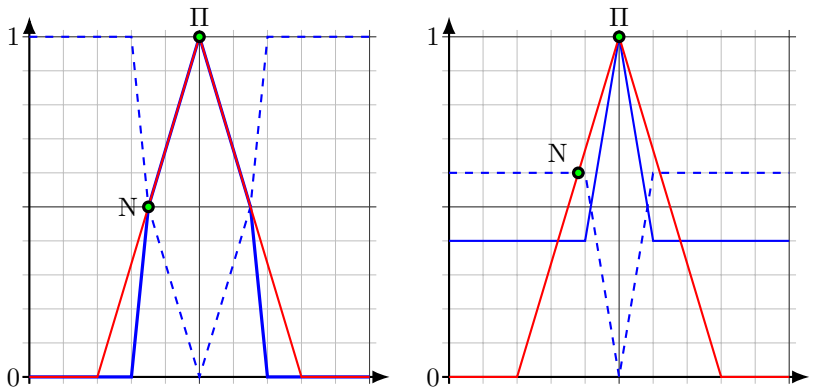
So what information is conveyed in Π and N? As was already established, Π and N converge to a degree of suitability when evaluating a datum that is certain. Dually, a fully uncertain datum will always result in $\Pi = 1$, $N = 0$. It thus seems that the difference $\Pi - N$ is an indicator of how uncertain the underlying datum is rather than to which degree it satisfies c.

The possibility-necessity couple form a (weak) total order over the results, sorting them by descending necessity first and descending possibility second. This order is sound for regular (Boolean) queries, but whether this is still true for the unclear semantics of the extension towards the evaluation of flexible queries is yet to be validated. In other words, does the same sorting strategy produce meaningful results? Perhaps the reason that the answer to this question is unclear is because of an underlying, fundamental challenge: how should one compare a certainly mediocre property to an uncertain but possibly very good one? The only correct answer to this question is: *"it depends"*. In critical applications, one might prefer a suboptimal yet predictable solution over one that is maybe better but also possibly worse. In other cases, one might choose to rely on the most plausible outcome to compare alternatives or be specifically hunting for objects that are uncertain. Thus really the question we should be answering is: can we achieve such semantically rich ranking strategies using these possibility-necessity degrees?

Let us investigate this by using an example. Imagine a set of entities and a query regarding one attribute that aims to achieve a ranking such that entities whose attribute value certainly satisfies the criterion (to some degree) are always ranked above those with an attribute value that possibly satisfies the criterion to a lesser degree (or not at all). In other words, the entities should be sorted according to their minimal degree of suitability, which could be considered a pessimistic evaluation because the worst case is used to compare them. Consider two example entities, as displayed in Figure 3.1. By computing Π and N for both, it is immediately clear that the desired ranking can not be derived. Option (b) seems better in every way (equal Π, higher N and smaller difference $\Pi - N$), despite the fact that option (a) is certain to take a value that at least satisfies c somewhat while option (b) can take an infinite amount of values that do not satisfy c at all.

3.2 Suitability Distributions

The suggested approach of using possibility and necessity degrees to represent the evaluation of a flexible criterion on possibilistic data is too limited. The semantics of these degrees are not clear and some desirable rankings of underlying data can not be attained from just these indicators. Furthermore, it can only be applied on possibilistic uncertainty models. Our objective is to find an alternative representation of the evaluation of a flexible criterion on uncertain



(a) Π = 1, N = 0.5 (b) Π = 1, N = 0.6

Figure 3.1: Two possiblistic properties and their computed possibility/necessity degrees. The properties (denoted by π) are drawn in blue ($1-\pi$ in dashed blue), the criterion (denoted by μ) in red.

data. This representation should have unambiguous semantics that are easy to interpret and should enable us to rank entities in intuitively meaningful ways. We will start our argument on possibilistic uncertainty, but it will be shown that our proposed approach can also be applied on other uncertainty models.

We start by revisiting the proposition that the correct representation is a possibility distribution over degrees of satisfaction. In what follows we will give a step-by-step procedure to construct these distributions and then present some of their properties. Next, we will discuss their semantics and show how they can be used to rank data in different ways rather than presuming that there is a single, true order. It is up to the user to define their tolerance towards uncertainty, which results in a specific order for their use case.

3.2.1 Definition

We will briefly repeat the nomenclature that will be used frequently in the coming sections. Recall that $\pi : A \to [0,1]$ is a *possibility distribution* reflecting the uncertainty regarding the value that attribute a takes. Hereby, for any $v \in A$, $\pi(v)$ represents the *possibility* that a takes v as value. Assume further that $\mathrm{c} : A \to [0,1]$ is a *criterion* over a. Hereby, for any $v \in A$, $\mathrm{c}(v) \in [0,1]$ denotes the degree to which v *satisfies* c (or else: to which v is *preferable* or *suitable*). The higher $\mathrm{c}(v)$, the more preferable v. With these concepts at hand, we wish to derive a belief function $s_{a,\mathrm{c}} : [0,1] \to [0,1]$ that expresses the uncertainty regarding the degree to which a satisfies c. Such a belief function will be called a *suitability distribution*.

Definition 1. *A suitability distribution is defined by a possibility distribution* $s_{a,\mathrm{c}} : [0,1] \to [0,1]$ *that models the uncertainty regarding the degree to which a*

specific entity's value for a satisfies c. *For each degree of satisfaction* $\sigma \in [0,1]$, $s_{a,\mathrm{c}}(\sigma)$ *denotes the possibility that* a *takes a value* $v_i \in A$ *that satisfies* c *to degree* σ *(i.e.* $\mathrm{c}(v_i) = \sigma$*).*

In essence, Definition 1 states that a suitability distribution is a mapping that associates each possible degree γ to which a can satisfy c with a degree of possibility. This degree of possibility can be obtained as follows. On the one hand, the uncertainty over a is given by a possibility distribution π over A. On the other hand, the criterion c expresses a functional relation between values in A and degrees of satisfaction. In order to construct a functional connection between degrees of satisfaction and degrees of belief, we intend to use (the inverse of) c to derive which values satisfy the criterion to a specific degree, to then determine how possible these values are by testing them in π. We will then aggregate those degrees of possibility to arrive at the overall possibility that a takes any domain value such that it satisfies c to that specific degree. Doing this for every possible degree of satisfaction, we obtain the full relation between degrees of satisfaction and degrees of belief. This procedure is shown schematically in Figure 3.2 and is explained and discussed in detail in what follows.

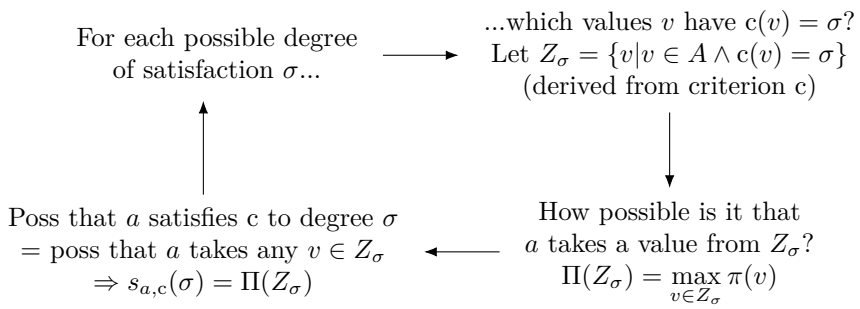


Figure 3.2: Schematic approach of constructing a suitability distribution.

3.2.2 Construction

Because it is uncertain which value a takes, it is uncertain to what degree c is satisfied. The idea is to perform a case-by-case analysis of each possible degree σ to which c might be satisfied, depending on what value a takes. For each σ, we use c to find all values that satisfy c to degree σ. Consider therefore a partitioning of A into classes Z_σ imposed by c such that all values from a single class are equally preferable:

$$Z_\sigma = \{v \in A | \mathrm{c}(v) = \sigma\}$$

Note that if the range of c is not finite, computing each Z_σ explicitly is impossible as there would be an infinite amount of partition classes corresponding to the

infinite variations in graded preference. However, it will be shown that they can be expressed analytically. For each Z_σ, we can compute the possibility that a takes any value from Z_σ. This can be viewed as the possibility that c is satisfied to the degree σ. Taking into account the definition of a possibility measure (and by extension a possibility distribution), this possibility is equal to the maximum of possibilities assigned to values in Z_σ. As such, we have found that:

$$s_{a,\mathrm{c}}(\sigma) = \max_{v \in Z_\sigma}(\pi(v))$$

This expression holds for every $\sigma \in [0, 1]$ and as such describes the analytical relation of the possibility that a satisfies c for any σ. Note further that, by construction, $s_{a,\mathrm{c}}(\sigma)$ is a possibility degree. Because this is true for every σ, $s_{a,\mathrm{c}}$ is a possibility distribution. The entire process is exemplified in Figures 3.3 and 3.4.

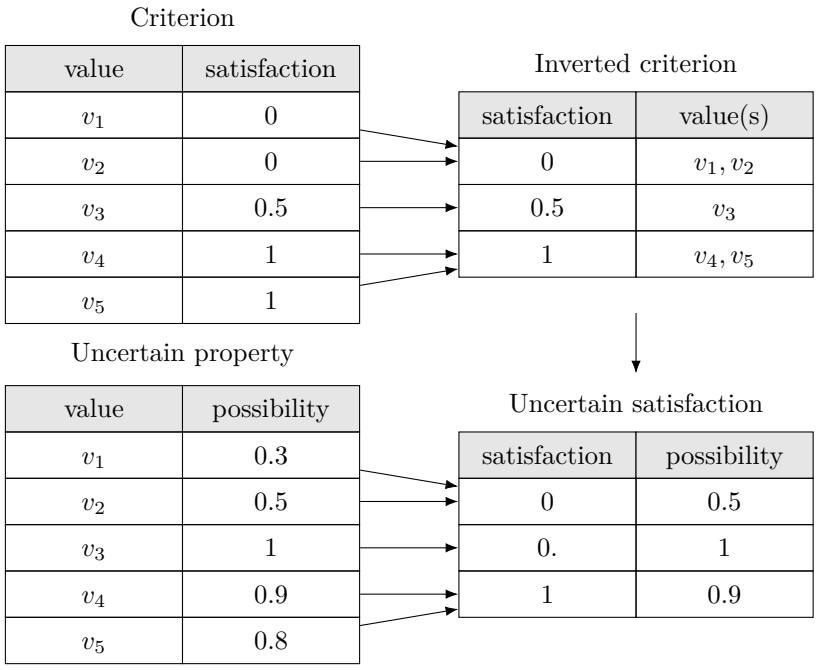


Figure 3.3: Example construction of a suitability distribution over a possibilistically uncertain attribute.

A suitability distribution as defined in Definition 1 denotes the *possibility* for each degree of satisfaction. However, it should be noted that the framework in which uncertainty is expressed, is not bound to be possibility theory. For instance, it is possible to infer a suitability distribution that models the *necessity* for each degree of satisfaction. Consider therefore the dual suitability distribution $s^*_{\pi,\mathrm{c}} : [0, 1] \to [0, 1]$ that reflects the necessity of a satisfying c to degree σ.

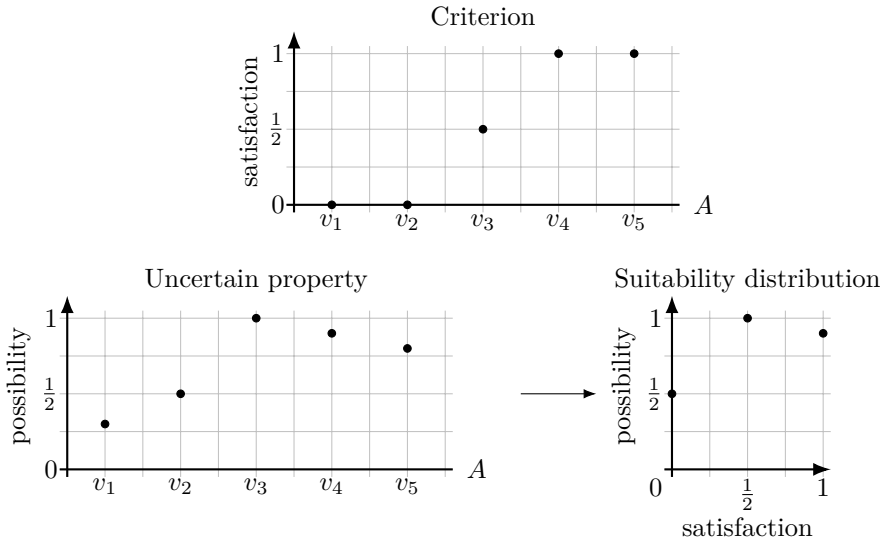


Figure 3.4: Graphical visualization of the construction of a suitability distribution.

The construction of $s^*_{\pi,\mathrm{c}}$ is similar to that of $s_{a,\mathrm{c}}$. We start by defining:

$$s^*_{\pi,\mathrm{c}}(\sigma) = \mathrm{N}(Z_\sigma)$$

Unlike with possibility, we can not easily compute the necessity that a takes a value from Z_σ. Recall that, by definition, a necessity measure is a confidence measure that satisfies $\mathrm{N}(\emptyset) = 0$, $\mathrm{N(A)} = 1$ and $\mathrm{N}(A \cap B) = \min(\mathrm{N}(A), \mathrm{N}(B))$, but there is no rule to determine $\mathrm{N}(A \cup B)$ from $\mathrm{N}(A)$ and $\mathrm{N}(B)$ However, we can fall back on the relation between N and Π:

$$\mathrm{N}(A) = 1 - \Pi(\bar{A})$$

Here, $\bar{A}$ is the complement of A (i.e. $A \setminus A$) denoting the event "a takes a value that is *not* in Z_σ". From this we find:

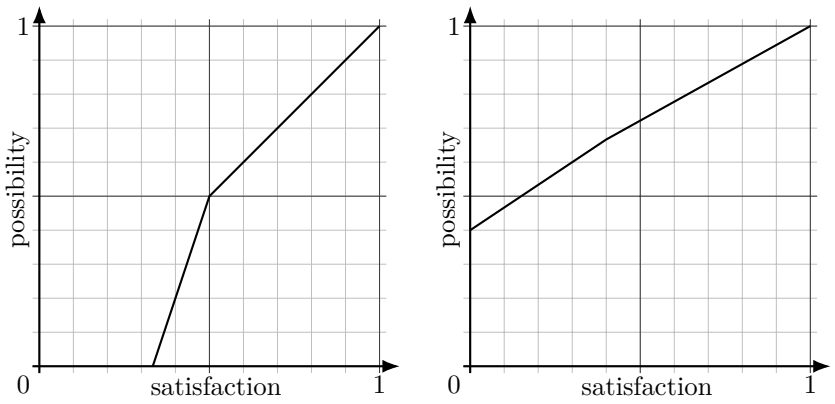
$$\begin{aligned} s^*_{\pi,\mathrm{c}}(\sigma) &= 1 - \Pi(\bar{Z_\sigma}) \\ &= 1 - \max_{v \notin Z_\sigma}(\pi(v)) \end{aligned}$$

As expected, $s^*_{\pi,\mathrm{c}}(\sigma)$ denotes the complement to 1 of the possibility that a takes a value that is *not* in Z_σ. From this, it follows that only when $s_{a,\mathrm{c}}$ is unimodal (i.e. $\exists!\sigma_1 \in [0,1], s_{a,\mathrm{c}}(\sigma_1) = 1$), $s^*_{\pi,\mathrm{c}}$ is not zero everywhere. Indeed, only if there is but a single fully possible degree of satisfaction, it has a certain necessity, but as soon as there are multiple different fully possible degrees of satisfaction, every degree of satisfaction has zero necessity. This corresponds to the findings of Dubois and Prade in [14]. As $s^*_{\pi,\mathrm{c}}$ is only circumstantially non-zero for (at

most) a single degree of satisfaction, we argue that $s^*_{\pi,c}$ is less informative than $s_{a,c}$. Going forth, we will not study it further.

Regardless of the framework in which uncertainty is expressed, a suitability distribution is a representation of how the uncertainty about the value of a property a affects the outcome of a criterion defined over a. From this compact representation, we can derive a lot of valuable information. For instance, in a setting of risk analysis, we typically want to account for the worst (resp. best) possible degree of satisfaction. This information is obtained by $\min\{\gamma \in [0,1] \mid s_{a,c}(\gamma) > 0\}$ (resp. $\max\{\gamma \in [0,1] \mid s_{a,c}(\gamma) > 0\}$). Alternatively, in an optimistic setting, we might want to know the lowest (resp. highest) degree of satisfaction that is fully plausible. Again, this information is easily obtained by $\min\{\gamma \in [0,1] \mid s_{a,c}(\gamma) = 1\}$ (resp. $\max\{\gamma \in [0,1] \mid s_{a,c}(\gamma) = 1\}$). Suitability distributions also allow us to answer a lot of interesting questions regarding the property and the criterion it was subjected to. For example, we can compute the answer to questions like: "Can a satisfy c fully?" and "What is the possibility that c is at least κ satisfied?". These questions are answered by respectively "$s_{a,c}(1) > 0$" and "$\max_{\gamma>\kappa} s_{a,c}(\gamma)$". Note that these questions can be considered as regular criteria on the uncertain property "satisfaction", which can be expressed using a possibility and a necessity, which can both be computed directly from $s_{a,c}$. For example, the necessity that c is at least κ satisfied equals $1 - \max_{\gamma\leq\kappa} s_{a,c}(\gamma)$. One could even evaluate a suitability distribution using a flexible criterion, e.g. "Which properties likely have a high degree of satisfaction?". The result thereof would again be a suitability distribution.

Let us revisit the example from Figure 3.1 to see what rankings we can derive using suitability distributions. Figure 3.5 shows the suitability distributions for both properties. These clearly reflect that property (b) might not satisfy c at all, whereas option (a) satisfies c at least 1/3.

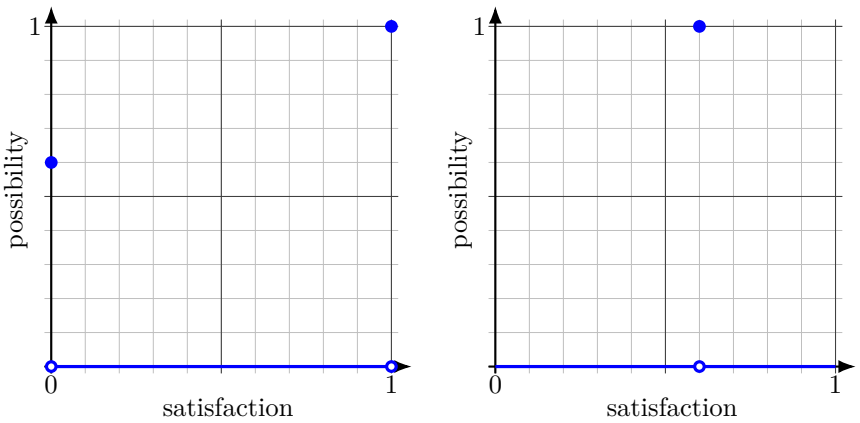


(a) $s_{a,c}$ (b) $s_{b,c}$

Figure 3.5: The suitability distributions for the two properties from the first example.

3.2.3 Special cases

We will now show how suitability distributions do in fact generalize the results of both flexible querying on certain data and regular querying on uncertain data.



(a) Boolean query on uncertain data. (b) Flexible query on certain data.

Figure 3.6: Visualization of special cases.

Let us examine how using a Boolean criterion b to evaluate a impacts the suitability distribution. A Boolean criterion partitions the domain into two classes: satisfactory (i.e., $\gamma = 1$) and not satisfactory (i.e., $\gamma = 0$). Consequently $s_{a,b}(\gamma) = 0, \forall \gamma \in]0, 1[$. Figure 3.6a illustrates an example thereof. Such suitability distributions can be summarized by two numbers: the possibility for satisfaction and the possibility for non-satisfaction. While not so surprising, this result reveals that a suitability distribution collapses to a Possibilistic Truth Value (PTV)[15, 16, 17] in the Boolean case. In addition, this also means that a suitability distribution for Boolean criteria collapses to possibility and necessity degrees in the sense of Prade et al. [23]. This can be shown as follows:

$$\begin{aligned}
\Pi(b) &= \max_v \min(\pi(v), \mu_b(v)) \\
&= \max(\min_{v \in Z_0}(\pi(v), 0), \min_{v \in Z_1}(\pi(v), 1)) \\
&= \max_{v \in Z_1}(0, \pi(v)) \\
&= \max_{v \in Z_1} \pi(v) \\
&= s_{a,\mathrm{c}}(1)
\end{aligned}$$

This shows that the possibility for satisfaction (i.e. $s_{a,\mathrm{c}}(1)$) is equal to the possibility degree that the property satisfies the criterion (i.e. $\Pi(b)$). Similarly, the possibility for non-satisfaction (i.e. $s_{a,\mathrm{c}}(0)$) can be seen as the complement

to 1 of the necessity degree that the property satisfies the criterion (i.e. N(b)):

$$\begin{aligned}
\Pi(\bar{b}) &= \max \min_{v}(\pi(v), \mu_{\bar{b}}(v)) \\
&= \max \min_{v}(\pi(v), 1 - \mu_{b}(v)) \\
&= \max(\min_{v \in Z_0}(\pi(v), 1), \min_{v \in Z_1}(\pi(v), 0)) \\
&= \max_{v \in Z_0}(\pi(v), 0) \\
&= \max_{v \in Z_0} \pi(v) \\
&= s_{a,\mathrm{c}}(0) \\
&= 1 - \mathrm{N}(b)
\end{aligned}$$

From this it should be clear that suitability distributions are in essence a generalization of the concepts of possibility and necessity degrees, which coincide with the generalized concepts introduced by Dubois and Prade for regular queries. Note that, in this case, these indicators are true possibility and necessity degrees and also satisfy all of their properties. In other words, either $s_{a,\mathrm{c}}(1) = 1$ or $s_{a,\mathrm{c}}(0) = 1$.

As a second special case, consider the suitability distribution resulting from the evaluation of a flexible criterion on a certain property. Assume a is certain to take value v_1. As this is certain, the degree to which a satisfies the criterion c is certain. Namely, it is equal to $\mathrm{c}(v_1)$. Thus $s_{a,\mathrm{c}}(\mathrm{c}(v_1)) = 1 \wedge s_{a,\mathrm{c}}(\nu) = 0, \forall \nu \in [0,1] \setminus \{c(v_1)\}$. As expected, the certainty regarding the value of a is reflected by the certainty of the suitability distribution. The suitability distribution is shown in Figure 3.6b. In this case, computing the suitability distribution is basically computing the degree of satisfaction of the property's value. This illustrates that suitability distributions not only generalize possibility and necessity degrees but also flexible querying on certain data, effectively generalizing both.

3.3 Properties of Suitability Distributions

Probably the most important property of suitability distributions is that they maintain the belief model of the underlying uncertain attribute. That is to say, if the uncertainty regarding an attribute value is modelled using possibility theory, the suitability distributions obtained by evaluating flexible criteria will be possibility distributions. Consequently, suitability distributions immediately inherit a lot of interesting properties. Moreover, there are certain correlations between the properties of the distribution of the uncertain value and those of their resulting suitability distributions. In what follows, we will discuss the most important of these properties and provide a proof for their correctness. For the remainder of this section, let π_X denote the possibility distribution of uncertain property X (the identifier X may be omitted if it is clear which property is being referred to) and a criterion regarding that property be denoted by c_X. Let the suitability distribution resulting from the evaluation of π_X under c_X be

noted as s_X. Furthermore, let $supp_\pi$ and $core_\pi$ denote the support and core of the fuzzy set that corresponds to the possibility distribution π.

Property 1. *The suitability distribution of a property represented by a normalized possibility distribution is also normalized.*

Proof. If π is normalized, $\exists v_1 \in A : \pi(v_1) = 1$. Furthermore, the degree of satisfaction for v_1 can be found from c and equals $\mathrm{c}(v_1)$. We then have $s_{a,\mathrm{c}}(\mathrm{c}_{v_1}) = \Pi(Z_{\mathrm{c}_{v_1}}) = 1$, which shows that $s_{a,\mathrm{c}}$ is indeed normalized. □

Regarding to information conveyed by a possibility distributions, one can consider the following ordering. A possibility distribution π_X is said to be *at least as specific* as π_Y if $\forall v \in A : \pi_X(v) \leq \pi_Y(v)$ [11]. This leads to the presentation of property 2.

Property 2. *If π_X is at least as specific as π_Y, then s_X is at least as specific as s_Y, regardless of* c.

Proof. Given $\forall v \in A : \pi_X(v) \leq \pi_Y(v)$, then:

$$\begin{aligned} \forall \kappa \in [0,1] : s_X(\kappa) &= \max_{v \in Z_\kappa} \pi_X(v) \\ &\leq \max_{v \in Z_\kappa} \pi_Y(v) \\ &\leq s_Y(\kappa) \end{aligned}$$

□

Property 2 implies that the order induced by specificity about attribute a is transferred to a specificity order on the satisfaction of a. The same transfer is observed for *unimodality*.

Property 3. *The suitability distribution for a criterion evaluated over a unimodal possibility distribution is again unimodal.*

Proof. By contradiction: assume π is unimodal (i.e. $\exists! v_p \in A : \pi(v_p) = 1$) and s is not unimodal (i.e. $\exists \rho_1, \rho_2 \in C : \rho_1 \neq \rho_2$, such that $s(\rho_1) = s(\rho_2) = 1$). Recall that $s(\rho) = \max_{v \in Z_\rho} v$. Thus, $s(\rho_1) = 1 \Rightarrow \exists v_1 \in Z_{\rho_1} : \pi(v_1) = 1$ and $s(\rho_2) = 1 \Rightarrow \exists v_2 \in Z_{\rho_2} : \pi(v_2) = 1$. Because $Z_{\rho_1} \cap Z_{\rho_2} = \emptyset$ (they are subsets generated by a partition), it follows that $v_1 \neq v_2$. This would imply the existence of two different values that are fully possible, which is in contradiction to our assumption that π is unimodal. □

Note that a non-unimodal possibility distribution can lead to a unimodal suitability distribution if $core_\pi \subset Z_\rho$ for some degree of suitability ρ. However, if a suitability distribution is not unimodal, then the original possibility distribution of the property π is not unimodal.

Property 4. *For a single criterion* c*, two properties can only result in identical suitability distributions if $\forall Z_\kappa : \max_{v \in Z_\kappa} \pi_X(v) = \max_{v \in Z_\kappa} \pi_Y(v)$. We say they are* c*-equivalent.*

Proof. Given that, under c, $s_X(\kappa) = s_Y(\kappa) : \forall \kappa \in [0,1]$, then:

$$\begin{aligned} s_X(\kappa) &= \max_{v \in Z_\kappa} \pi_X(v) \\ &= \max_{v \in Z_\kappa} \pi_Y(v) \\ &= s_Y(\kappa) \end{aligned}$$

Indeed, s_X and s_Y can only be identical if every subset Z_κ from the partitioning on A imposed by c is equally possible for X and Y. □

Property 5. *If at least one unpreferable value is possible (i.e.,* $\mathrm{c}(v) = 0 \wedge \pi(v) > 0$*), there is a non-zero possibility for non-satisfaction.*

Proof.

$$\begin{aligned} \exists v_0 \in Z_0 \subseteq A : \pi(v_0) > 0 \\ \Rightarrow s(0) = \max_{v \in Z_0} \pi(v) \\ \geq \pi(v_0) \\ > 0 \end{aligned}$$

□

Property 6. *If at least one preferable value is possible, there is a non-zero possibility for full satisfaction.*

Proof. Analogue to the proof of Property 5 but with Z_1 instead of Z_0. □

3.4 Interpretation and ranking of Suitability Distributions

Now that we have introduced suitability distributions as a concept, we will discuss their usability. More specifically, we will take a look at what information can be derived from a suitability distribution and how they, as generalizations of degrees of suitability, can be used to rank query evaluation results. We will limit this discussion to possibilistic suitability distributions, because they have properties that are particularly useful when it comes to ranking.

3.4.1 Semantics of the suitability distribution

A suitability distribution models uncertainty about the suitability of a value against a given criterion. Any uncertainty regarding the suitability degree stems from uncertainty regarding the underlying attribute value being evaluated, though even uncertain values can lead to certain suitability degrees in the event that the user is indifferent towards the region containing all possible values. For example, it does not matter if John's age is specified as "between 6

and 10" when the user is only interested in mature subjects: John is definitely not suitable. Suitability distributions with a convex membership function are most interesting, as they can usually be translated back to a linguistic term that is indicative of the represented vague suitability degree ("good", "bad", "mediocre"...).

A suitability distribution contains information regarding uncertainty. More precisely, the area of the suitability surface can be used to quantify the amount of uncertainty, akin to standard deviation and variance in a probabilistic setting. This area will practically (i.e. when using the unit interval to denote both the possibility domain and the suitability domain) be quantifiable by a numerical value between 0 and 1. Even if different domains are used, it is certain that the uncertainty that is present in a suitability distribution will always be bound between 0 and a certain upper bound that is straightforward to compute.

This information regarding uncertainty is extremely valuable. Based on such a quantification of uncertainty, one is able to express a preference for entities with particular attribute values and moreover with a limited amount of uncertainty. Otherwise, uncertainty quantification can be valuable when the goal is to uncover which entities are potentially very suitable, but too uncertain to trust in analyses, meriting further investigation or data collection.

3.4.2 Ranking using suitability distributions

Though typical suitability distributions are fuzzy intervals, other forms of uncertainty (e.g. John is either around 18 or at least 42 years old) also exists. Suitability distributions from such uncertain values will usually correspond to a fuzzy set with a non-convex shape function. Nonetheless, we will limit ourselves to studying fuzzy intervals as non-convex shape functions can be broken down into either a disjunction of convex parts or covered by a convex hull.

Several studies have already been devoted to the comparison of fuzzy numbers [12, 2, 22], but there is no real consensus in this area of research because the semantics of fuzzy sets play a fundamental role when it comes down to how they should be interpreted and, consequently, compared [13]. As we are dealing with uncertain suitability degrees, we need to find a way to compare them in a way that is meaningful. Regular suitability degrees have a clear interpretation in the sense that a higher value means a more suitable entity. Similarly, the worst possible entity corresponds to a suitability distribution that only has mass on 0 and the best possible entity to a distribution that only has mass on 1. Generic suitability distributions describe an area of suitability between these bounds, indicating some form of graded suitability.

Under the assumption that we construct a flexible preference that is a flawless representation of our preferences, then, when comparing two entities, a strictly larger suitability degree, however small, would imply that the corresponding entity is strictly more suitable to our needs than the other. We consider this a meaningful ranking strategy for suitability degrees. Suitability distributions represent a set of possible suitability degrees for a possibly uncertain value and, as such, imply a range of suitability for the corresponding entity.

In some cases, suitability distributions can be ranked unambiguously, despite being uncertain. Consider for example the three entities (a), (b) and (c) represented by their respective suitability distributions shown in Figure 3.7. Here, the fact that there is uncertainty regarding their precise degrees of suitability is irrelevant as in any case the order is clear: (a) is strictly better than (c) (regardless which values they both take), which is always strictly better than (b) (it is certainly worse than the worst possible case for (c)).

In essence, these distributions can be ranked unambiguously because no matter which attribute values the underlying entities actually take, they would always be put in the same order according to the resulting suitability degrees.

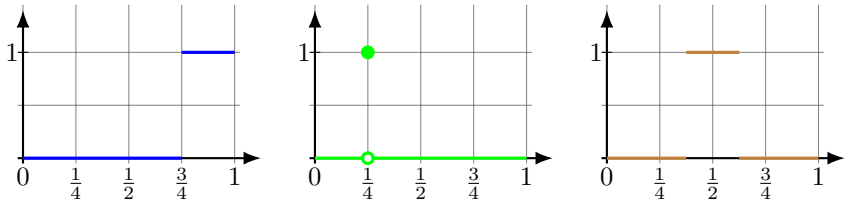
Unfortunately, there are also many situations wherein the order is not clear: as soon as the intersection of the supports of at least two suitability distributions contains more than a single value, it is no longer guaranteed that one entity is *always* better than the other. If we were to compute all possible rankings, we would notice that they are not all equivalent. One could try to quantify the possibility associated to each possible ranking and though this is, according to the authors, the only truly correct solution, such an approach quickly becomes unfeasible for large problems as the complexity of this solution grows exponentially with both the amount of distributions and the degree to which they are uncertain. Some of the research on this particular topic is described in [7, 18, 19].

Instead, we suggest a few heuristical approaches that allow us to compare suitability distributions by translating them to a single, representative number.

3.4.3 Defuzzification strategies

As the name implies, defuzzification aims to turn a fuzzy value into an un-fuzzy value. Defuzzification can be applied to suitability distributions in order to make it easier to compare them to each other so as to obtain a global ranking of a set of entities.

There are different ways to perform defuzzification, and different approaches can (and typically will) result in different rankings. Regardless of which approach is used, defuzzification will come with a loss of information, not only



(a) $s_{a,c}$ (b) $s_{b,c}$ (c) $s_{c,c}$

Figure 3.7: Example suitability distributions that are straightforward to compare.

due to the fact that we are mapping an entire function to a single number, but also due to so-called *collisions*. A collision describes the event where two different suitability distributions are defuzzified onto the same numerical value. When distributions collide, they can no longer be distinguished from one another. However, collisions are somewhat predictable in the sense that certain defuzzification strategies will always result in collisions of the same types of distributions. Keeping this in mind will help us predict the impact of collisions, which will in turn help us choose between stategies in particular situations.

In what follows, we will discuss several defuzzification strategies, broadly categorized into two types: those that quantify uncertainty and those that result in an indicator of overall suitability.

Defuzzifying with the intent to quantify uncertainty

Suitability distributions convey information about the amount of uncertainty regarding the suitability of an entity regarding a specific query. There are several ways to quantify this information, and we will discuss some of them here. Typically, we want to quantify uncertainty with the purpose of comparing entities, though it can be a goal in an of itself to rank entities based on their overall uncertainty. Technically, it is possible to rank entities according to uncertainty by simply rephrasing the query criteria to map uncertainty rather than domain values to suitability. As a result, one would obtain a suitability degree per entity that is representative of uncertainty, which can then be used for sorting purposes. However, suitability distributions make it possible to obtain similar rankings even when the original query was constructed towards suitability without even considering that attribute values can be uncertain up front.

Quantification using the suitability area Perhaps the most intuitive way to quantify the uncertainty of a suitability distribution is to compute the area underneath the curve. This surface is an indicator of uncertainty for fuzzy sets in general, but because suitability distributions are expressed over the suitability domain, it is guaranteed that the area underneath *any* suitability distribution will be between 0 and a fixed upper bound (1 in case the unit interval is used as suitability domain). This does not just allow us to rank entities based on this quantification, but also gives insight into the absolute amount of uncertainty of each entity regarding a specific query. This way of defuzzifying suitability distributions will result in collisions between all distributions that have the same surface. This can be counter-intuitive as the shape of the suitability distribution plays a large part in its surface calculation, and a triangular suitability distribution will have to span a lot more possible values in the suitability domain than a rectangular (uniform) suitability distribution in order to result in the same quantified degree of uncertainty. Arguably, the former distribution is more uncertain, as it spans a wider range of possible suitability degrees. In addition, it should be noted that this approach should only be applied to uncertain suitability intervals (i.e. uncertain suitability degrees whose distribution is a convex membership function), as otherwise it can happen that certain and par-

ticular uncertain suitability distributions collide by being mapped to 0 (like, for example, suitability distributions that result from the evaluation of a Boolean criterion on an uncertain attribute value).

Quantification using the support of the distribution As an alternative approach to quantify uncertainty, one could compute the difference between the minimal and maximal possible suitability degree from the distribution. Again due to the nature of the domain of suitability distributions, this defuzzification strategy is guaranteed to result in a number between 0 and 1. In the absence of uncertainty, minimal and maximal suitability are identical, resulting in a quantification of 0. However, a whole family of suitability distributions can result in the maximal uncertainty quantification of 1, even if there is only a single fully possible suitability degree. This defuzzification strategy will result in collisions between distributions with a similar core, but this time will not distinguish between triangular and rectangular distributions, as they span the same range of possible values.

Quantification using the support and core As a final strategy, a combination of the previous approaches can be used, thereby trying to strike a balance between the area underneath the suitability distribution and the amount of possible suitability degrees it spans. As such, we propose to compute not only the width of the support of the suitability distribution, but also that of the core of the suitability distribution, and averaging both. As a result, rectangular suitability distributions will have a maximal amount of uncertainty with respect to their support, and triangular distributions a minimal amount (in case of a regular average: half the width of the support). Using this approach, uncertain values will still be the only ones that map to 0, but the uniform distribution is again the only one that will result in a quantification of 1.

Suitability-focussed defuzzification

Suitability-focussed defuzzification strategies are those that translate a suitability distribution to a suitability degree. These are "traditional" in the sense that they follow the definition that defuzzification is the process of mapping a fuzzy number to a regular number from the same domain. Intuitively, the number resulting form defuzzifying a fuzzy number should be part of its support (i.e. the chosen value should be a possible value).

Center of mass defuzzification A first defuzzification strategy that satisfies this intuition is projection onto the abscissa of the center of mass x_g [26] of the suitability area. This can be viewed as a sort of weighted average of all possible outcomes, where the graded possibility of each value is used as its weight:

$$x_g = \frac{\int_0^1 x f(x) \mathrm{d}x}{\int_0^1 f(x) \mathrm{d}x}$$

Semantically, the center of mass can be viewed as an indicator of the "expected" degree of suitability, taking into account all possible outcomes. As such, it portrays an attitude towards uncertainty which is generally robust to outliers that have little possibility. This approach will reliably map unsuitable entities to 0 and perfect entities to 1. The more uncertain, the closer to 0.5 the defuzzified value will be. Indeed, a maximally uncertain value (the uniform suitability distribution) maps to exactly 0.5. Note that the region near 0.5 is also where mediocre entities will be mapped to. As such, the collisions for this defuzzification strategy are typically between (highly) uncertain and mediocre entities. One could argue that this behavior is a *feature*, as it will guarantee that the almost certainly good and bad attributes will be mapped to respectively high and low degrees of suitability, and that the middle zone is purposefully used to collect the remaining (rather uncertain and certainly mediocre) results. From that point of view, the center of mass approach can be chosen purposefully in order to reliably separate the best and worst entities from the mediocre and uncertain ones. This is not unusual because users are typically looking for reliable and good entities. We say that this strategy denotes an attitude that exhibits *caution*, as it acts as a kind of filter for uncertain entities.

Optimistic defuzzification Another way to defuzzify suitability distributions is by taking the *maximal* possible degree of suitability that is associated with a degree of possibility larger than 0. In this case, the result of the defuzzification corresponds to the event that the attribute takes the best possible value regarding the specified preferences, however unlikely that is. Choosing for this approach illustrates an optimistic attitude by believing in the best possible case. Alternatively, it could be seen as a greedy attitude, aggressively assuming the best possible case while neglecting the fact that reality may very well be less optimal. It would be less aggressive to limit oneself to the suitability distribution's core instead, and to take the best fully possible value. In fact, one can use any particular α-cut to place a limit on minimal required possibility. The closer α lies to 0, the greedier the approach. Choosing for a specific α-cut is comparable (though also different) to relying on confidence intervals in statistics. However, suitability distributions allow one to specify both a confidence interval and an α-cut, granting more flexibility.

Optimistic attitudes are typical for prediction systems which assume "normal conditions", such as GPS-based routing software that predict time in the assumption that no sudden accidents happen which could influence travel time. Note that this defuzzification strategy always produces a suitability degree from the support of the suitability distribution, even for those with a non-convex membership function. Collisions caused by this approach make it impossible to distinguish between one the one hand certainly good and on the other hand uncertain but possibly good entities. In order to ensure that certain entities are placed before uncertain entities that can reach the exact same degree of suitability, one could break ties based on a quantification of uncertainty. For example, the area underneath the suitability distribution could, in this context,

be considered as an indication of "how much worse" the entity could technically be, essentially quantifying the possible risk of choosing this entity. As such, one would rank by the defuzzified suitability first, and break ties by ascending risk. One could argue that this defuzzification strategy is not in line with the philosophy behind fuzzy logic, as it can lead to large distances between defuzzified values that are considerably close to each other. For example, two entities whose suitabilities are certain and known to be equal to x and $x + \delta$ (where δ can be any arbitrarily small but strictly positive number), will be separated by all entities whose suitability is uncertain but maximally equal to $x + \delta$. In order to avoid this "crisp", un-fuzzy behavior, we propose a different strategy. The idea is to compute the optimistically estimated suitability degree and to quantify the amount of uncertainty, and to combine both values in a sort of weighted average. Hereby it is assumed that the quantified uncertainty is first mapped to a degree of suitability y. Usually, y will be maximal for entities with no uncertainty and minimal for maximally uncertain suitability distributions. In any case, the quantified amount of uncertainty under optimistic defuzzification depends on the maximal suitability degree x itself, and should be bound between $[0, x]$. As such, y is always maximally 1 (there is no uncertainty) and the higher x, the lower y can be. Only fully uncertain suitability distributions (uniform across the entire suitability domain) can lead to $y = 0$. We propose the following formula to compute the final value x_f:

$$x_f = x - \gamma(1 - y) \tag{3.5}$$

where $\gamma \in [0, 1]$ can be used to tune the impact of uncertainty (0 implying indifference and 1 implying most severe). Indeed, when $\gamma = 0$, the outcome is always x, regardless of how uncertain x is. When there is no uncertainty ($y = 1$), the final score x_f is equal to x, regardless of γ. In all other cases, the optimistic estimate x is lowered by an indication of the amount of uncertainty $(1-y)$, multiplied by the impact factor. If $\gamma = 1$, a fully uncertain suitability distribution is mapped to 0 ($x = 1, y = 0$). A γ-value of 0.5 will result in a balance with equal weight for both optimism and uncertainty, somewhat approximating the center of mass defuzzification. Lowering γ increases the impact of the optimistic assumption.

Pessimistic defuzzification Dually, one could also defuzzify by taking the *minimal* possible value that is associated with a degree of possibility larger than 0, assuming the worst possible case. Consequently, this defuzzification strategy denotes a pessimistic attitude. This might be valuable when the outcome of the decision is of critical importance and there is no room for error. As such, it can be seen as an attitude of risk aversion. Again, one might also choose a slightly less pessimistic attitude by choosing the minimal fully possible value, e.g. the lowest suitability degree with degree of possibility equal to 1. As such, very unlikely bad outcomes are purposefully ignored, again assuming "normal circumstances". Also here, collisions hide uncertainty, but again this can be mitigated similarly to the previous approach. However, in this case, ties should

be broken by descending (rather than ascending) risk, as uncertainty from a pessimistic point of view would imply that the entity could actually be better than estimated. Similarly to the optimistic case, this approach is not very fuzzy, and we can introduce a more fuzzy alternative. In this case, minimal uncertainty would be mapped to minimal suitability. Indeed, from a pessimistic angle, uncertainty is a good thing as it can only improve the suitability of an entity, so more uncertainty is better. Letting z denote the suitability derived from the quantified uncertainty of the suitability distribution, we have:

$$x_f = x + \gamma z \tag{3.6}$$

where, again, $\gamma \in [0, 1]$ is the impact factor balancing suitability versus uncertainty. Note that, if $x = 1$ under pessimistic defuzzification, there can be no uncertainty and hence $z = 0$ and $x_f = 1$, regardless of γ. Again, a γ-value of 0.5 will map the uniform distribution (in which case $z = 1$ to indicate a maximal possible improvement in suitability) to a score of 0.5. More generally, a higher value for γ will result in a higher impact of the uncertainty on the defuzzified suitability value.

Reflection

We have discussed how to defuzzify a suitability distribution to either a suitability degree or a quantification of uncertainty, but it certainly seems most informative to have both indicators. This is not a new idea, it is very similar to how parameters are used to describe entire probability mass functions. For example, a Gaussian distribution is fully defined by merely its expected value and its standard deviation, two indicators that are very closely related to the center of mass on the one hand and the width of the (α0.05 cut) support of the suitability distribution on the other. However, we should question the reliability of the estimated suitability grade when the quantified uncertainty is very high, just like we would be (rightfully) skeptical about the computed expected value of a uniform distribution. Indeed, if a suitability distribution is in fact uniform, then we can turn it into any result we want by choosing a specific defuzzification strategy and tuning its parameters, though the uncertainty will always be maximal. This would seem to imply that the suitability degree is of no importance when the uncertainty is maximal, so why bother with defuzzifying to a specific value anyway? This extends beyond uniform distributions; the significance of a specific suitability grade diminishes as the uncertainty increases.

However, there actually exist situations where the available information leads to an exact suitability degree and a simultaneous overall degree of uncertainty. One such situation can be found in the context of time-sensitive measurements. Consider that, at a given time, a certain time-sensitive attribute is recorded, for example the address of a friend. As time passes, you grow out of touch, but after some years you decide to invite your friend to your wedding party. However, you are not certain about whether or not the address on record is still correct. Theoretically, we could digitize this scenario using a couple of values: the original address, the moment at which you wrote it down and a parameter

that estimates the rate at which people move (the rate of "decay"), a process that could probably be approximated by an exponential ditribution. Note that this is similar to how arrival rates are modelled in queueing theory. In the context of evaluating uncertain data, these could lead to a specific suitability degree (based on the recorded value) and a quantified uncertainty (based on the rate of decay and the time that has passed since the initial measurement).

This example is a bit artificial, as in the digital age that we live in, it is very easy to keep connected through social media and you can reach out to your friend at any time to verify their address (in other words: remeasuring the data is fast, cheap and easy). However, it is just an example to show how information that is stored in databases can become outdated as the actual value changes from the one that was once recorded. Unlike the address of a friend, a value that is stored in a database might not be so easy to measure, for obvious reasons. In fact, many data are actually time-sensitive (more generally, they are context-sensitive, but for this discussion, we will limit ourselves to changes that occur due to the passing of time), though this is usually ignored and what is stored in the database is considered correct. Off of this observation, we base our motivation for an approach with a couple of grades that denote suitability and uncertainty. Nonetheless, if an entire suitability distribution is given, then a defuzzification to a suitability grade, a quantification of uncertainty and even a combination of both still comes at a rather large loss of information.

Bipolar defuzzification

We have discussed how the semantics of quantified uncertainty depends on the defuzzification strategy that is used (pessimistic, optimistic, ...). This is particularly undesirable for aggregation purposes, as we will see in the next chapter. In order to mitigate this, we propose a final defuzzification strategy that can be considered an extension that can be applied to all of the previously introduced strategies.

Rather than defuzzifying to a suitability grade and a single indicator of uncertainty, we will instead use two indicators of uncertainty: one indicative of the amount of possible suitabilities *lower* than the defuzzified suitability grade and one indicative of the amount of possible suitability *higher* than the defuzzified suitability grade. The resulting couple of uncertainty grades could be seen as a bipolar uncertainty indicator, with one of the grades having a negative and the other having a positive connotation. The fact that these indicators have fixed semantics is the driving force behind this approach.

Note that this extension can be applied for any combination of strategy for calculating a suitability grade and method to quantify uncertainty from the distribution. Essentially, rather than quantifying the entire distribution, the distribution is split in two parts by the defuzzified suitability grade, and both parts are quantified separately.

Example Consider defuzzifying the trapezoidal suitability distribution given in Figure 3.8 using the center of mass strategy. This results in the defuzzified

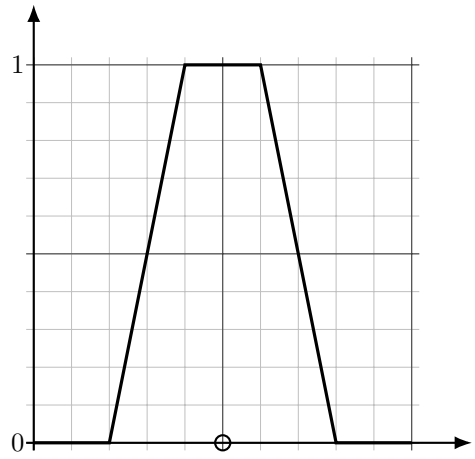


Figure 3.8

suitability grade of 0.5. Quantifying the uncertainty as the area under the distribution, we find 0.4 If we apply the extension, we would find $(0.2, 0.2)$ to be the bipolar uncertainty grade, as there is an equal amount of uncertainty mass to the left of the defuzzified suitability grade as there is to the right. If we, alternatively, use a soft pessimistic defuzzification (based on the core), we find that the defuzzified suitability grade of the distribution is equal to 0.4 instead and that, though the area under the distribution is still 0.4, the bipolar uncertainty grade now equals $(0.1, 0.3)$. This reflects that the center of mass defuzzification leads to a suitability grade that could be worse but also better, whereas the pessimistic defuzzification leads to a suitability grade that could actually be much better and not much worse.

3.5 Towards broader applications

In the above, we have introduced the concept of suitability distributions under the initial assumption that information about a was given in the form of a possibility distribution. In addition, we have assumed that both degrees of satisfaction and degrees of possibility are expressed using a real number in the unit interval.

In this section, we further develop the use of other theories of uncertainty and at the same time introduce more general scales of preference. This leads to the following, more general, definition of a suitability distribution:

$$s_{u,\mathrm{c}} : S \to B, \sigma \to s_{u,\mathrm{c}}(\sigma) = \operatorname*{G}_{v \in Z_\sigma} (u(v))$$

where S is a bounded lattice denoting a range of degrees of satisfaction, B is a bounded lattice (not necessarily equal to S) denoting a range of degrees of belief,

$u : A \to B$ is a distribution of uncertainty, associating each value $v \in A$ with a degree of belief $u(v)$ that v is the actual value of a and finally G the operator to meaningfully aggregate degrees of belief from B. Consider for example $S = [-1, 1]$ (where -1 denotes dissatisfaction, 0 neutrality and 1 satisfaction), $B = [0, 1]$ and u a probability distribution, e.g. a Gaussian distribution. In that case, G is replaced by a summation for discrete properties and an integration for continuous properties. Alternatively, B could be an ordinal scale used to express levels of confidence, u an ordinal possibility distribution [9, 11] and S an ordinal scale of degrees of satisfaction. In what follows, we will demonstrate that suitability distributions can be used in all these cases.

3.5.1 Discrete Probabilistic Uncertainty

Consider the case where the uncertainty over a is given by a discrete probability distribution p. In this case, the belief assigned to Z_σ must reflect the *probability* that a takes a value v in Z_σ. The additivity of a probability measure allows us to replace G by the $\sum$ operator. Substituting this in the definition of $s_{\mathrm{p,c}}$, we get:

$$s_{\mathrm{p,c}}(\sigma) = \sum_{v \in Z_\sigma} (\mathrm{p}(v))$$

Example. Consider a fair 6-sided die. The outcome of throwing the die can be predicted using a uniform distribution, where each outcome is equally probable:

$$\forall i \in \{1, 6\} : \mathrm{p}(i) = \frac{1}{6}$$

Assume the following (discrete) preference for "high" outcomes (i.e. high outcomes are more satisfying than low ones):

$$\begin{aligned} \mathrm{c}(1) &= 0 \\ \mathrm{c}(2) &= 0 \\ \mathrm{c}(3) &= 0 \\ \mathrm{c}(4) &= 0.5 \\ \mathrm{c}(5) &= 1 \\ \mathrm{c}(6) &= 1 \end{aligned}$$

Then the resulting suitability distribution would be:

$$\begin{aligned} s_{\mathrm{p,c}}(0) &= \mathrm{p}(1) + \mathrm{p}(2) + \mathrm{p}(3) = \frac{3}{6} \\ s_{\mathrm{p,c}}(0.5) &= \mathrm{p}(4) = \frac{1}{6} \\ s_{\mathrm{p,c}}(1) &= \mathrm{p}(5) + \mathrm{p}(6) = \frac{2}{6} \end{aligned}$$

Note that $s_{\mathrm{p,c}}$ is again a probability distribution as the sum of all outcomes equals 1. It can easily be seen that this will always be true as the suitability distribution is by construction nothing more than a summation of all probabilities according to a specific partitioning into satisfaction classes imposed by c. This illustrates that suitability distributions preserve the framework of uncertainty.

3.5.2 Continuous Probabilistic Uncertainty

Consider the case where the uncertainty over a is given by a continuous probability distribution p, called a *density*. So far, we have been able to use the criterion to find all values resulting in a specific degrees of satisfaction. For discrete properties, this is done exhaustively case-by-case by considering all values. However, this is impossible for continuous properties as the mapping is infinite. Instead, we must rely on analytical expressions describing all relations. Considering c is the analytical expression associating values with degrees of satisfaction, we can use its inverse c^{-1} to find all values corresponding to a degree of satisfaction. Of course this requires that c^{-1} exists, which implies c must be invertible. However, because a criterion might (and most likely will) reflect that some values are equally satisfactory, c might not always be analytically invertible as a whole. To mitigate this, we require that c can be written as a piecewise function that consists of either *strictly monotonic* or *constant* sub-functions. Though this might seem restrictive, it is almost always true in practice, as criteria are commonly expressed using linear correlations between values and degrees of satisfaction due to their simplicity and intuitiveness. Indeed, the most commonly used shape for membership functions, used to model criteria, are trapezoids.

The key to approaching c as explained above is that each sub-function can be treated separately, leading to a collection of *partial* suitability distributions. The entire suitability distribution is the function sum of all partial suitability distributions. Assume by convention that all values of the domain not covered by a sub-function are considered to be mapped to $\inf(S)$, denoting they are undesirable.

Constant sub-functions. Let us first examine constant sub-functions. Any such criterion denotes a range of values $[x, y] \subseteq A$ that are all equally preferable: $\forall v \in [x, y] : \mathrm{c}(v) = \kappa$, reflecting indifference towards specific values from $[x, y]$. Thus, the statements "a takes a value from $[x, y]$" and "c is satisfied to degree κ" are interchangeable. From probability theory we know that the probability that the attribute takes any value from $[x, y]$ equals the mass of the density function in that range, thus:

$$\text{Prob[c is satisfied to degree } \kappa] = \int_x^y \mathrm{p}(t)\,\mathrm{d}t$$

Because all values in $[x, y]$ lead to satisfaction degree κ, there is no way a can satisfy c to a degree other than κ. As such, the partial suitability distribution

$s^{(i)}_{a,\mathrm{c}}$ is 0 everywhere, except in κ. We express this using the Dirac-impulse, which leads to:

$$s^{(i)}_{\mathrm{p,c}}(\tau) = k * \delta(\tau - \kappa)$$

where $k = \int_a^b \mathrm{p}(x)\,\mathrm{d}x$ is a scaling factor which safeguards that the mass of the partial suitability distribution is equal to the mass of p over $[x, y]$.

Strictly monotonic sub-functions. Let us now look at strictly monotonic sub-functions. Let $f : [x, y] \to [\alpha, \beta]$ be a strictly monotonic function reflecting an ordering of values in $[x, y] \subseteq A$ according to degrees of satisfaction in $[\alpha, \beta] \subseteq S$. If f is increasing then $f(a) = \alpha$ and $f(b) = \beta$, and $\forall s, t \in [x, y], s < t : f(s) < f(t)$. The probability that a then takes a value v_k from $[x, y]$ so that the criterion is satisfied to degree *at most* $\kappa = f(v_k) \in [\alpha, \beta]$ equals:

$$U(v_k) = \int_x^{v_k} \mathrm{p}(t)\,\mathrm{d}t$$

where U is the cumulative probability function of p. Because f is strictly monotonic, it is invertible, so $f^{-1} : [\alpha, \beta] \to [x, y]$ exists and $f^{-1}(\alpha) = a$, $f^{-1}(\beta) = b$ and $f^{-1}(\gamma) = v_k$. Substituting this connection between attribute values and degrees of satisfaction gives:

$$U(f^{-1}(\gamma)) = \int_{f^{-1}(\alpha)}^{f^{-1}(\gamma)} \mathrm{p}(x)\,\mathrm{d}x$$

from which it can be seen that U can be written as a function of degrees of satisfaction. Moreover, $U(f^{-1}(\gamma))$ can be interpreted as the probability that the attribute takes any value so c is satisfied to a degree of at most γ. Differentiating this expression leads to the partial suitability density $s^{(i)}_{\mathrm{p,c}}$:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\gamma} U(f^{-1}(\gamma)) &= \frac{\mathrm{d}}{\mathrm{d}\gamma} \int_{f^{-1}(\alpha)}^{f^{-1}(\gamma)} \mathrm{p}(x)\,\mathrm{d}x \\ &= \mathrm{p}(v)\end{aligned}$$

which, due to the invertibility of f and given $\tau = f(v)$, can be written as

$$\begin{aligned}\mathrm{p}(v) &= \mathrm{p}(f^{-1}(\tau)) \\ &= s^{(i)}_{\mathrm{p,c}}(\tau)\end{aligned}$$

This reflects that the unit probability that the criterion is satisfied to degree τ equals the unit probability that the attribute takes the value v that corresponds to that particular degree of satisfaction. It also shows that the suitability distribution can be analytically expressed as a simple combination of the uncertainty model (i.e., p) and the inverse of the preference model (i.e., f^{-1}). Its domain is $[f^{-1}(a), f^{-1}(b)] = [\alpha, \beta]$ and its range is B. An identical reasoning can be applied to strictly monotonically decreasing sub-functions.

Combined. In the above, we have chosen to take a step-by-step approach based on the decomposition of c into sub-functions. From each sub-function, a resulting partial suitability distribution s_i can be found as described, effective in the range of the sub-function. The total suitability distribution can be found by summing all these partial suitability distributions:

$$s_{\mathrm{p,c}}(\sigma) = \sum_i s^{(i)}_{\mathrm{p,c}}(\sigma)$$

Note that this requires that all partial suitability distributions share the same domain (S), which is not necessarily true: constant sub-functions only have mass for a single degree of satisfaction and strictly monotonic sub-functions are only defined on $[\alpha, \beta] \subseteq S$. However, any partial suitability distribution s_i can easily be expanded:

$$s'^{(i)}_{\mathrm{p,c}}(\sigma) = \begin{cases} s^{(i)}_{\mathrm{p,c}}(\sigma) & \sigma \in [\alpha, \beta] \\ 0 & \sigma \in S \setminus [\alpha, \beta] \end{cases}$$

Note that a partial suitability distribution in general only covers a portion of the total probability mass of the density p. This is true as soon as c consists of more than one sub-function. As such, a partial suitability distribution is not a density function. The entire suitability distribution however, as the sum of all partial suitability distributions which form a complete partition of the probability mass, is a density function, in analogy to the discrete case.

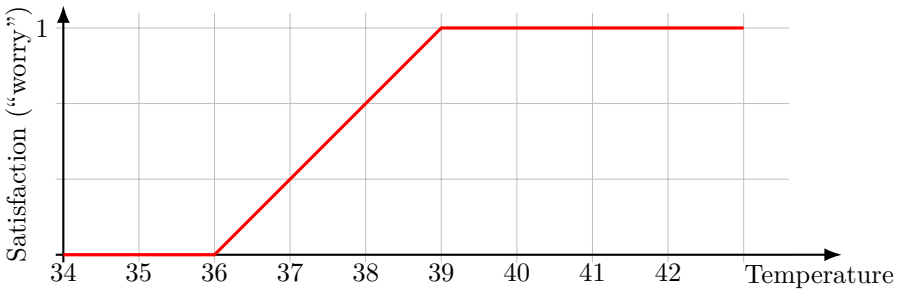


Figure 3.9: Example of a suitability distribution over a probabilistically uncertain property.

Example. Consider an example where a temperature is measured using measuring equipment which is known to be error-prone according to a Gaussian distribution with a standard deviation of 2 degrees Celsius. Assume further we are trying to identify if we should be worried concerning a subject's body temperature, using a flexible criterion reflecting the degree to which we find certain temperatures worrying, modeled by the membership function visualized in figure 3.9. A measurement has resulted in a reading of 36.5 degrees Celsius.

Thus, the uncertainty regarding the subject's real temperature can be written as:

$$T(t) = \mathcal{N}(t|36.5, 2) = \frac{1}{2}\phi\left(\frac{t - 36.5}{2}\right)$$

with ϕ the standard normal distribution, $\mathcal{N}(x|0, 1)$. To conclude if there is reason to worry about the subject given this reading, we construct the suitability distribution of the measurement according to the criterion. To that end, we decompose the criterion into three sub-functions:

- f_0 = any temperature below 36: no reason for worrying.
- f_1 = all temperatures above 39: extremely alarming.
- f_2 = a temperature between 36 and 39: a linear increase in worry with increasing temperature.

The first sub-function is constant, and as a result its corresponding partial suitability distribution will be a Dirac impulse (assume here we denote satisfaction with variable w, interpretable as a degree of worry):

$$\begin{aligned} s^{(0)}(w) &= \delta(w) \int_{-\infty}^{36} T(t)\,\mathrm{d}t \\ &= \delta(w) \int_{-\infty}^{-1/4} \phi(x)\,\mathrm{d}x \end{aligned}$$

The second sub-function is also constant, and analogously will lead to the following partial suitability distribution:

$$s^{(1)}(w) = \delta(w - 1) \int_{5/4}^{+\infty} \phi(x)\,\mathrm{d}x$$

The last sub-function is linear, and as the range of the criterion in this interval is the entire unit interval $[0, 1]$, the domain of the resulting partial suitability distribution will also be $[0, 1]$. From f_2 we know how temperature t and satisfaction/worry w are correlated:

$$w = \frac{t - 36}{3} \Leftrightarrow t = 3w + 36$$

Substituting this in T, the distribution expressing the uncertainty over the temperature, we find the partial suitability distribution:

$$\begin{aligned} s^{(2)}(w) &= T(3w + 36) \\ &= \mathcal{N}\left(w|\frac{1}{6}, \frac{2}{3}\right) \\ &= \frac{3}{2}\phi\left(\frac{3w - 1/2}{2}\right), w \in [0, 1] \end{aligned}$$

The total suitability distribution can now be found by summing the partial suitability distributions. It is shown visually in Figure 3.10. It immediately reflects our uncertainty, though it shows it is more likely there is no reason to worry than there is. However, the nonzero probability for maximal worry (indicated by the scaled Dirac impulse, denoted by an upwards arrow) might be sufficient reason to perform further examinations of the subject. We can also see that, apart from the probability mass that was aggregated into Dirac impulses, the remaining density function is indeed part of a (rescaled and translated) Gaussian distribution.

It can easily be verified that the result is normalized. Indeed, the total probability equals 1 (Φ being the cumulative distribution function of ϕ):

$$\begin{aligned}
&\int_{-\infty}^{+\infty} s^{(0)}(w) + s^{(1)}(w) + s^{(2)}(w)\,\mathrm{d}w \\
&= \int_{-\infty}^{-1/4} \phi(x)\,\mathrm{d}x + \int_{5/4}^{+\infty} \phi(x)\,\mathrm{d}x + \int_0^1 s_2(w)\,\mathrm{d}w \\
&= \Phi(-0.25) + [1 - \Phi(1.25)] + \int_0^1 \frac{3}{2}\phi\left(\frac{3w - 1/2}{2}\right)\,\mathrm{d}w \\
&= \Phi(-0.25) + [1 - \Phi(1.25)] + \int_{-1/4}^{5/4} \phi(x)\,\mathrm{d}x \\
&= \Phi(-0.25) + [1 - \Phi(1.25)] + [\Phi(1.25) - \Phi(-0.25)] \\
&= 1
\end{aligned}$$

Figure 3.10: The resulting suitability distribution of the example illustrating continuous probabilistic uncertainty.

Discretization. In practice, continuous variables often are discretized as the concept of probability is much more comprehensible than that of density. Typically, this is done by splitting the domain into equal-width buckets. This form of intuitive discretization is very common and relies on the definition of a unit, denoting the width of the buckets. Consider for example the discretization of time into seconds, temperature into degrees, age into years, mass into grams,

and so on. Even distributions are often discretized (think of confidence intervals and α-cuts). Of course, the unit of discretization depends on the application and the measuring equipment used, which is often referred to as the granularity of discretization. Regardless, it is safe to assume that it will often be acceptable to use the far simpler discrete approach than the continuous approach.

3.5.3 Qualitative preference and uncertainty

Due to the maxitivity instead of additivity of possibility measures, one can use possibility distributions in an ordinal fashion. As a final scenario in which we develop the concept of suitability distributions, we consider the case where both uncertainty and preference are expressed in a purely qualitative manner. More specifically, it is assumed that both uncertainty and preference are expressed on an ordinal scale. Consider a discrete property for which $A = \{a, b, c, d\}$. Let

$$B = \{impossible, unlikely, plausible, likely\}$$

be an ordinal scale of belief and $\succ$ be a strict order denoting "more plausible than". Assume $likely \succ plausible \succ unlikely \succ impossible$. Let there be a mapping $m : B \to \mathbb{R}$ such that $\forall b_i, b_j \in B : b_i \succ b_j \Rightarrow m(b_i) > m(b_j)$ Using m we may derive $\max(b_i, b_j)$ from $\max(m(b_i), m(b_j))$. Note that the precise image of m is not important as long as the aforementioned holds. Let the uncertainty regarding a be represented by an ordinal possibility distribution π:

$$\begin{aligned}\pi(a) &= unlikely\\ \pi(b) &= plausible\\ \pi(c) &= impossible\\ \pi(d) &= likely\end{aligned}$$

Furthermore, let

$$S = \{unacceptable, acceptable, preferable\}$$

be an ordinal scale of satisfaction. Completely analoguous to the above, assume there is a strict order relation on these degrees of satisfaction (in the order in which they are defined) and that there exists a mapping to a numerical representation such that this order is translated to the greater-than relation. Let our preferences towards the values the property can take be modeled by c:

$$\begin{aligned}\mathrm{c}(a) &= acceptable\\ \mathrm{c}(b) &= preferable\\ \mathrm{c}(c) &= preferable\\ \mathrm{c}(d) &= unacceptable\end{aligned}$$

The resulting suitability distribution s is then equal to:

$$\begin{aligned}s(unacceptable) &= likely\\ s(acceptable) &= unlikely\\ s(preferable) &= plausible\end{aligned}$$

whose interpretation is immediately clear. This example highlights that suitability distributions can be applied in combination with ordinal frameworks for uncertainty and satisfaction. Note that this can also be applied to continuous properties.

3.6 Summary and future research opportunities

In this chapter, we have shown the intricacies of applying flexible criteria on uncertain data and we have subsequently introduced a technique that produces semantically rich results under circumstances for which we have argued they pose few constraints in realistic situations. Moreover, the proposed approach is a true generalization of both flexible querying on traditional data and traditional querying on uncertain data, and can thus be applied in all combinations, unlike other current solutions. The result of the approach yields a suitability distribution, a model for the uncertainty regarding the degree to which the attribute value in question is suitable for the specified flexible criterion. We have shown how suitability distributions can be computed for two common frameworks for uncertainty modeling: possibility and probability theory, both for discrete and continuous cases. We have given an overview of interesting properties these distributions have. Furthermore, we have shown how they can be compared and how they might be aggregated. Towards automated ranking, we argue that one must choose a strategy expressing a subjective attitude towards uncertainty, similar to how a criterion expresses a preference towards specific domain values. However, it is possible to perform the evaluation without loss of information and without the need to specify such an attitude, if ranking is not the purpose. Essentially, the act of ranking is decoupled from the evaluation process, which is also different from existing approaches, wherein the user must be aware of uncertainty before specifying their initial preferences.

Towards future research, it would be interesting to investigate the applicability of suitability distributions on other, more extended models for uncertainty, including type-2 fuzzy sets, interval-valued fuzzy sets, Atanassov's intuitionistic fuzzy sets [1] and hesitant fuzzy sets [27, 24].

Bibliography

[1] Krassimir T Atanassov. "Intuitionistic fuzzy sets". In: *Fuzzy sets and Systems* 20.1 (1986), pp. 87–96.

[2] Christophe Billiet, Antoon Bronselaer, and Guy De Tré. "A Comparison Technique for Ill-known Time Intervals". In: *Proceedings of the IEEE International Conference on Fuzzy Systems*. Vancouver, Canada, 2016, pp. 1963–1969.

[3] P. Bosc, M. Galibourg, and G. Hamon. "Fuzzy querying with SQL: Extensions and implementation aspects". In: *Fuzzy Sets and Systems* 28.3 (1988), pp. 333–349. ISSN: 01650114. DOI: 10.1016/0165-0114(88)90039-5.

[4] Patrick Bosc and Olivier Pivert. "Fuzzy queries against regular and fuzzy databases". In: *Flexible query answering systems*. Springer, 1997, pp. 187–208.

[5] Patrick Bosc and Olivier Pivert. "SQLf: a relational database language for fuzzy querying". In: *IEEE transactions on Fuzzy Systems* 3.1 (1995), pp. 1–17.

[6] M Cayrol, H Farreny, and H Prade. "Fuzzy pattern matching". In: *Kybernetes* 11.2 (1982), pp. 103–116.

[7] Graham Cormode, Feifei Li, and Ke Yi. "Semantics of ranking queries for probabilistic data and expected ranks". In: *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*. IEEE. 2009, pp. 305–316.

[8] Robin De Mol, Antoon Bronselaer, and Guy De Tré. "Evaluating flexible criteria on uncertain data". In: *Fuzzy Sets and Systems* (2017).

[9] Didier Dubois and Henri Prade. "Formal Representations of Uncertainty". In: *Decision-Making Process: Concepts and Methods* (2009), pp. 85–156.

[10] Didier Dubois and Henri Prade. "Possibility theory , probability theory and multiple- valued logics : A clarification". In: *Annals of Mathematics and Artificial Intelligence* 32 (2001), pp. 35–66. ISSN: 1012–2443. DOI: 10.1023/A:1016740830286.

[11] Didier Dubois and Henri Prade. "Practical Methods for Constructing Possibility Distributions". In: *International Journal of Intelligent Systems* 31.2 (2015), pp. 215–239. ISSN: 08848173. DOI: 10.1002/int.

[12] Didier Dubois and Henri Prade. "Ranking fuzzy numbers in the setting of possibility theory". In: *Information sciences* 30.3 (1983), pp. 183–224.

[13] Didier Dubois and Henri Prade. "The three semantics of fuzzy sets". In: *Fuzzy sets and systems* 90.2 (1997), pp. 141–150.

[14] Didier Dubois and Henri Prade. "Unfair coins and necessity measures: Towards a possibilistic interpretation of histograms". In: *Fuzzy Sets and Systems* 10.1-3 (1983), pp. 15–20. ISSN: 01650114. DOI: 10.1016/S0165-0114(83)80099-2.

[15] Gert De Cooman. "Possibility theory 1: The measure- and integral-theoretic groundwork". In: *International Journal of General Systems* 25.4 (1997), pp. 291–323.

[16] Gert De Cooman. "Possibility theory 2: Conditional possibility". In: *International Journal of General Systems* 25.4 (1997), pp. 325–351.

[17] Gert De Cooman. "Possibility theory 3: Possibilistic independence". In: *International Journal of General Systems* 25.4 (1997), pp. 353–371.

[18] Ihab F Ilyas, George Beskales, and Mohamed A Soliman. "A survey of top-k query processing techniques in relational database systems". In: *ACM Computing Surveys (CSUR)* 40.4 (2008), p. 11.

[19] Jeffrey Jestes et al. "Semantics of ranking queries for probabilistic data". In: *IEEE Transactions on Knowledge and Data Engineering* 23.12 (2011), pp. 1903–1917.

[20] Janusz Kacprzyk and Andrzej Ziolkowski. "Database queries with fuzzy linguistic quantifiers". In: *IEEE transactions on systems, man, and cybernetics* 3.16 (1986), pp. 474–479.

[21] Cengiz Kahraman. *Fuzzy multi-criteria decision making: theory and applications with recent developments*. Vol. 16. Springer Science & Business Media, 2008.

[22] "Possibilistic evaluation of sets". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 21.3 (2013), pp. 325–346. DOI: 10.1142/S0218488513500177.

[23] Henri Prade and Claudette Testemale. "Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and Vague Queries". In: *Information Sciences* 143 (1984), pp. 115–143.

[24] Rosa M Rodríguez et al. "Hesitant fuzzy sets: state of the art and future directions". In: *International Journal of Intelligent Systems* 29.6 (2014), pp. 495–524.

[25] Valiollah Tahani. "A conceptual framework for fuzzy query processing: a step toward very intelligent database systems". In: *Information Processing & Management* 13.5 (1977), pp. 289–303.

[26] Tomohiro Takagi and Michio Sugeno. "Fuzzy identification of systems and its applications to modeling and control". In: *IEEE transactions on systems, man, and cybernetics* 1 (1985), pp. 116–132.

[27] Vicenç Torra. "Hesitant fuzzy sets". In: *International Journal of Intelligent Systems* 25.6 (2010), pp. 529–539.

[28] Zeshui Xu. *Uncertain multi-attribute decision making: Methods and applications*. Springer, 2015.

[29] Lotfi A Zadeh. "The concept of a linguistic variable and its application to approximate reasoning". In: *Information sciences* 8.3 (1975), pp. 199–249.

[30] Sławomir Zadrożny et al. "An overview of fuzzy approaches to flexible database querying". In: *Database Technologies: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications* 1 (2009).

Chapter 4

Aggregation and Suitability Distributions

4.1 Introduction

So far, we have introduced the suitability distribution as a way to represent the (uncertain) degree to which an (uncertain) attribute value satisfies a given criterion, discussed its semantics and shown how it can be defuzzified in different ways in the interest of sorting a set of entities. In order to obtain a single, objective and consistent ranking, it is required that the user not only formulates a suitability function over the domains of the attributes, but also an attitude towards uncertainty. We have shown how suitability distributions are not only flexible in the sense that they can reflect how different attitudes result in different rankings, but also that they can be used to answer questions regarding uncertainty in itself.

However, many systems rely on more than a single criterion when evaluating and comparing alternatives. Typically, there is a so-called *multi-criteria decision making* process or a multi-criteria query evaluation process, which involves different criteria and inter-criteria relations. Whereas the criteria denote preferences towards specific attributes' domain values, the inter-criteria relations represent logical connectives between the criteria. Specific platforms, called decision support systems, rely on fuzzy logic in order to be able to translate the decision logic of decision makers in a way that adequately reflects human reasoning. This is then used to compute a global suitability grade per entity. The overall purpose is to make it easier for the decision maker to get an overview of which entities satisfy their combined preferences.

Multi-criteria decision support is usually a two-step process: first a set of *elementary criteria* (i.e. criteria on particular attributes) are defined and evaluated, and second an *aggregation structure* is used to compute a single suitability grade per entity. The aggregation structure models an inter-criteria synergy, which corresponds to particular logic connectives. Examples of such connec-

tives are *all*, *any*, *most*, *some* and so on.

We will study if and how suitability distributions can be used in decision support systems. Our approach is to replace the elementary suitability grades with suitability distributions. The question we need to answer is how this will impact the mathematical backbone of the aggregation structure.

The remainder of this chapter is structured as follows. In section 4.2, we will first give detailed information on how aggregation is done traditionally. This overview will introduce basic aggregation operators but also more advanced, fuzzy aggregators, that are capable of modelling linguistic connectives such as "most", "some", "few"... and different ways to implement them. Then, in section 4.3, we show how fuzzy integration (one such implementation) can be adapted so that it can be used to implement partial absorption operators (a subset of these flexible aggregation operators). Afterwards, section 4.4 discusses the aggregation of suitability distributions. This discussion first explores the option to defuzzify before aggregating, and the advantages and disadvantages that come with it. Second, the focus is on aggregating suitability distributions directly, ranging from basic operators to fuzzy operators. Remaining research opportunities are summarized in section 4.5, which concludes the chapter.

4.2 Aggregation: Structure and Quantifiers

So far, we have focussed on evaluating a set of entities using a flexible criterion, with the intention to rank the entities from best to worst. As soon as at least two criteria are included in an evaluation process, ranking entities stops being a trivial task. Indeed, for each criterion, an entity is evaluated to a suitability grade representative of how well it satisfies the corresponding criterion, but according to which should the entities then be sorted? Similar to the case of sorting suitability distributions, there are some situations which are unambiguous and in which a total order is unarguably clear. For example, if entity r_1 has a suitability grade vector $(0.8, 1, 0.6)$ for three corresponding elementary criteria $\mathrm{c}_1, \mathrm{c}_2$ and c_3, then compared to entity r_2, which scores $(0.3, 0.5, 0.1)$ for the same criteria, it is better in every aspect. Unambiguously, r_1 is a better candidate than r_2. However, how does r_1 compare to r_3, defined by suitability grade vector $(1, 0.7, 0.8)$? Without specifying the relative importance of the criteria and their logical connectives, it is impossible to say which entity is better. In order to make a decision, however, we need to select one. Decision support systems try to be of assistance in this regard by using an aggregation structure to compute a single, global (suitability) suitability grade per entity.

4.2.1 Properties of aggregation structures

It is usually assumed that the result of aggregating several (elementary) suitability grades should be a suitability grade that lies between the minimum and maximum of its inputs. Moreover, if all inputs are equal, then the output should be equal to the inputs, too. As such, aggregation is typically implemented math-

ematically using a mean. The closer the outcome of an aggregator lies to the minimum, the more "conjunctive" it is said to be (dually "disjunctive" with respect to the maximum). The arithmetic mean is special because it is exactly as conjunctive as it is disjunctive and thus depicts a neutral form of aggregation. Aggregators that have a result between the arithmetic mean and the minimum (full conjunction) are said to be *partial conjunctions*. These embed the semantics of *simultaneity*, meaning that lower elementary suitability grades will have a larger negative impact on the outcome than higher elementary suitability grades will have a positive impact. An entity that satisfies only "most" criteria well will, through partial conjunction, be mapped to a mediocre global suitability grade. Similarly, aggregators whose result falls between the arithmetic mean and the maximum (full disjunction) are called partial disjunctions, which model *replaceability*. As the term suggests, replaceability implies that criteria can, to some grade, replace each other, representing a sort of redundancy. Under partial disjunction, it suffices that an entity satisfies no more than "some" of the criteria well in order to obtain a good suitability grade. Though aggregators can be categorized purely on how conjunctive (or disjunctive) they are, it is more meaningful to approach them from a semantic point of view [12]. For example, it is easier to understand that "at least 3 of the 5 criteria should be satisfied" than it is to interpret what it means when an aggregator is conjunctive to degee 0.8. The different frameworks that implement fuzzy aggregators differ mainly in the semantics that they support, and how precisely they can be fine-tuned [16].

In its simplest form, an aggregation structure corresponds to a single aggregator that expresses the inter-criteria synergy connecting all elementary criteria. However, aggregation structures can also be constructed hierarchically, first combining certain elementary criteria into an intermediate suitability grade, then aggregating these intemediary grades further with a separate aggregator. This nesting process can be repeated as many times as necessary until a single, global suitability grade is obtained. This makes it possible to model compound logic, like "either this criterion should be satisfied, or at least 2 of those 3 criteria should be".

4.2.2 Boolean aggregators

Commonly known are the *existential* and *universal* quantifiers, which denote that respectively *any* and *all* criteria must be satisfied. These also represent the Boolean inter-criteria connectives 'or' and 'and', respectively representing the full disjunction and the full conjunction.

In multi-valued logic, the conjunction is implemented using a *triangular norm* (t-norm) [23, 19]. A t-norm $\top$ must satisfy the following properties:

- $\top(a, b) \leq \top(c, d)$ if $a \leq c$ and $b \leq d$ (monotonicity)
- $\top(a, b) = \top(b, a)$ (commutativity)
- $\top(a, \top(b, c)) = \top(\top(a, b), c)$ (associativity)

- $\top(a, 1) = a$ (1 is the identity element)

Though there are different variations that realize the t-norm, the Gödel t-norm (also the *minimum t-norm*) is considered the standard for conjunction in fuzzy logic because it is the only one that satisfies the properties of an aggregation operator. In addition, it is also the strictest of the t-norms, which corresponds to full conjunction. The Gödel t-norm $\top_{min}$ essentially selects the minimum of its inputs:

$$\top_{min}(s_1, s_2, \ldots, s_n) = \min_i s_i = \bigwedge_{i=1}^{n} s_i$$

It can quickly be verified that the Gödel t-norm confirms classical logic by decoding `true` to 1 and `false` to 0, in which case the outcome indeed corresponds to that of the traditional, Boolean conjunction.

Some of the other t-norms (like the product and Łukasiewicz t-norms) formalize even stronger conjunctions. For example, for two identical elementary suitability grades 0.5 and 0.5 (reprensenting two semi-satisfied criteria), the conjunction under the different t-norms is computed as follows:

$$\top_{min}(0.5, 0.5) = \min(0.5, 0.5) = 0.5 \quad (4.1)$$

$$\top_{prod}(0.5, 0.5) = 0.5.0.5 = 0.25 \quad (4.2)$$

$$\top_{Luk}(0.5, 0.5) = \max(0, 0.5 + 0.5 - 1) = 0 \quad (4.3)$$

Though these t-norms could all be used to compute a global suitability grade, they will not be considered further as they do not respect the desired properties of aggregators. In addition, it can be seen that the final ranking that would be obtained by applying these alternative t-norms would largely be similar to that under the Gödel t-norm, albeit slightly rescaled (i.e. the *distance* between two outputs may be different depending on the t-norm that is used).

The classical disjunction can be realized by a *triangular conorm* $\perp$ (t-conorm or s-norm). T-conorms must satisfy the same four properties as t-norms with the exception that the identity element is 0, rather than 1. Similarly to the Gödel t-norm, the Gödel t-conorm (or maximum t-conorm) is considered the standard in fuzzy logic:

$$\perp_{max}(s_1, s_2, \ldots, s_n) = \max s_i = \bigvee i = 1^n s_i$$

4.2.3 Advanced aggregators

For most practical problems, it is rarely the case that precisely all or only one of the criteria should be satisfied, especially when the amount of criteria is large. Hence, we will need other aggregators than just t-norms and t-conorms. Another aggregator that is well known is the arithmetic mean:

$$s_g = \frac{\sum_{i=1}^{n} s_i}{n}$$

In order to be able to distinguish which criteria are more important than others, relative weights can be added to each of the criteria:

$$s_g = \frac{\sum_{i=1}^{n} w_i s_i}{\sum_{i=1}^{n} w_i}$$

These weights express relative importance, i.e. $w_1 = 2w_2$ implies that s_1 is twice as important as s_2. The weights are often normalized such that their sum equals 1, which makes it easier to derive the absolute impact of a certain weight and generally simplifies the formula (the denominator disappears). With these weights, it is possible to filter certain inputs by setting their weight to 0. The corresponding input will then have no impact on the outcome of the aggregator and the outcome would be the same as if that particular input has not included in the aggregation in the first place.

Simplified to the Boolean case, the arithmetic mean boils down to the fraction of criteria that are satisfied. In the fuzzy case, it gives an overview of the average elementary suitability grades. Unfortunately, when using the arithmetic mean in fuzzy situations, it is impossible to distinguish between entities that fully satisfy half of the criteria and entities that partially satisfy all criteria to grade 0.5. In situations where we need to do just that, we need to use a different aggregator (which will turn out to be a partial conjunction).

Partial aggregators (partial conjunctions and disjunctions) have been implemented through different mathematical approaches. In the following, the three leading approaches are summarized.

The Generalized Conjunction/Disjunction

Quantifiers such as 'at least X', 'about Y', 'most', 'few',... are too complicated to represent mathematically with a traditional weighted average. Dujmović studied the use of the power mean and found that he was able to link its outcome to different degrees of "andness" (simultaneity) by changing the value of the exponent p [15]:

$$s_g = \left(\frac{1}{n} \sum_{i=1}^{n} s_i^p \right)^{(1/p)}$$

As with a traditional average, there is also a weighted variant of the generalized mean:

$$s_g = \left(\sum_{i=1}^{n} w_i s_i^p \right)^{(1/p)}$$

In this form, it is assumed that the weights are already normalized such that they sum up to 1. Similarly to the traditional weighted average, the weights dictate the importance of the inputs, and assigning a weight of 0 to an input can be done in order to exclude that input from the calculations. Based on his definition of the generalized conjunction/disjunction (GCD), Dujmović implemented his own decision support system called Logic Scoring of Preferences (LSP) [12].

Dujmović categorizes aggregators into five classes: full conjunction ($\wedge$), partial conjunction ($\triangle$), arithmetic mean, partial disjunction ($\triangledown$) and full disjunction ($\vee$).

$$y = x_1 \diamondsuit \ldots \diamondsuit x_m = \begin{cases} x_1 \vee \ldots \vee x_m, & \alpha = 0, \omega = 1 \\ x_1 \triangledown \ldots \triangledown x_m, & 0 < \alpha < 0.5, 0.5 < \omega < 1 \\ (x_1 + \ldots + x_m)/m, & \alpha = \omega = 0.5 \\ x_1 \triangle \ldots \triangle x_m, & 0.5 < \alpha < 1, 0 < \omega < 0.5 \\ x_1 \wedge \ldots \wedge x_m, & \alpha = 1, \omega = 0 \end{cases}$$

In the above, α denotes "andness" and ω denotes "orness". Note that α and ω are tightly coupled and technically only of them is needed in order to be able to express all degrees of generalized conjunction/disjunction. Using the (weighted) power mean, α (and dually ω) is tied to the value of the exponent p. Particular values of p correspond to particular degrees of andness. A summary of 17 concrete levels, proposed by Dujmović in [13], is given in Table 4.1. This table also shows the output that is obtained by aggregating a fully satis-

Gradient	**Symbol**	**p**	α	**GCD(1, 0)**
Strongest Conjunction	C	$-\infty$	1	0.000
Very Strong Conjunction	C++	-9.060	15/16	0.000
Strong Conjunction	C+	-3.510	7/8	0.000
Medium Strong Conjunction	C+−	-1.655	13/16	0.000
Medium Conjunction	CA	-0.720	3/4	0.000
Medium Weak Conjunction	C−+	-0.148	11/16	0.000
Weak Conjunction	C−	0.261	5/8	0.070
Very Weak Conjunction	C−−	0.619	9/16	0.326
Neutrality	A	1	1/2	0.500
Very Weak Disjunction	D−−	1.449	7/16	0.620
Weak Disjunction	D−	2.018	3/8	0.709
Medium Weak Disjunction	D−+	2.792	5/16	0.780
Medium Disjunction	DA	3.929	1/4	0.838
Medium Strong Disjunction	D+−	5.802	3/16	0.887
Strong Disjunction	D+	9.521	1/8	0.930
Very Strong Disjunction	D++	20.63	1/16	0.967
Strongest Disjunction	D	$+\infty$	0	1.000

Table 4.1: Gradients of the partial conjunction/disjunction and an example of a GCD aggregator.

fied and a fully unsatisfied criterion. We can see that the partial conjunction produces a 0 output up to the "medium weak conjunction", despite the value of p changing. This range signifies a "hard" partial conjunction. We say any requirements combined by those aggregators are *mandatory*. Other variants of the partial conjunction (the weak partial conjunction and the very weak partial conjunction) are considered to combine inputs in a non-mandatory way, while

still rewarding situations where all inputs are high. The more we lean towards the full disjunction, the more orness plays a role and the more the global suitability grade increases up to the highest degree of satisfaction among the inputs, which is finally reached at the full, pure disjunction.

Note that there is no symmetry regarding a "hard" disjunction, only the most extreme value for p yields a true, full disjunction.

(Weighted) Ordered Weighted Averaging

Yager takes a different point of view with his ordered weighted averaging (OWA) approach [31, 30], in which a weight-generating function is used to model different degrees of conjunctivity.

To evaluate an OWA operator with a weight-generating function $f : [0, 1] \to [0, 1]$, one must first evaluate all criteria and sort them descendingly. Let $\mathbf{x}$ denote the sorted elementary suitability grades resulting from this first step, then the global suitability is calculated as follows:

$$s_g \overset{\text{OWA}}{=} \sum_{i=1}^{n} \mathbf{x}_i \left[f\left(\frac{i}{n}\right) - f\left(\frac{i-1}{n}\right) \right] \tag{4.4}$$

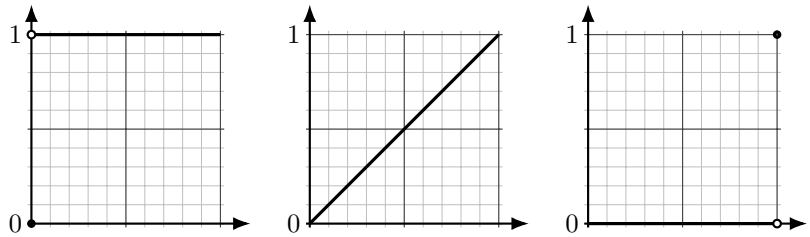
A function f can be used as a weight-generating function if it satisfies the following constraints:

$$f(0) = 0 \tag{4.5}$$
$$f(1) = 1 \tag{4.6}$$
$$\forall x, y \in [0, 1], x \leq y : f(x) \leq f(y) \tag{4.7}$$

In words, the weight-generating function must be monotonically increasing and be translated and scaled so it passes through both the origin and $(1, 1)$.

The weight-generating function defines the behavior of the OWA operator. It is possible to construct functions such that the OWA operator implements the disjunction, arithmetic mean and conjunction. These functions are shown in Figure 4.1. Consider, for example, the function depicted in Figure 4.1a. When used in formula 4.4, it is clear that x_1 will have weight 1 and that all remaining x_i will subsequently have weight 0. In other words, only the most satisfied criterion determines the output, and even defines it directly. Indeed, this is the behavior of the disjunction. Similarly, Figure 4.1c shows the conjunction and Figure 4.1b the arithmetic mean.

As mentioned earlier, partial conjunctions lie between the arithmetic mean and the (full) conjunction. An example of a weight-generating function that realizes a partial conjunction is shown in Figure 4.2a. It is easily verified visually that this function strikes a middle ground between the arithmetic mean and the conjunction, but let us examine an example to prove it is truly a partial conjunction.



(a) Disjunction (b) Arithmetic mean (c) Conjunction

Figure 4.1: Examples of functions that can be used to define the behavior of an OWA operator.

Example. Consider that, for some entity, the evaluation of three criteria leads to the following three elementary suitability grades: $(0.3, 1.0, 0.8)$. Using a conjunction, these values would aggregate to 0.3, whereas the arithmetic mean equals $(0.3+1.0+0.8)/3 = 0.7$. We will use the function $f(x) = x^2$ to represent a partial conjunction. First, $\mathbf{x} =< 1.0, 0.8, 0.3 >$ is constructed by sorting the elementary suitability grades. Then, we can compute s_g:

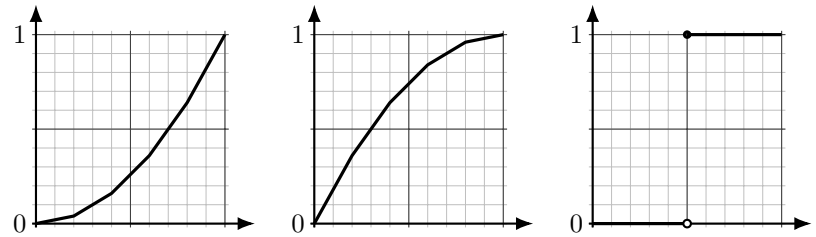
$$\begin{aligned} s_g &= 1\left(f\left(\frac{1}{3}\right) - 0\right) + 0.8\left(f\left(\frac{2}{3}\right) - f\left(\frac{1}{3}\right)\right) + 0.3\left(1 - f\left(\frac{2}{3}\right)\right) \\ &= \frac{1}{9} + \frac{3}{9}0.8 + \frac{5}{9}0.3 \\ &= \frac{49}{90} \end{aligned}$$

We indeed find that the result (roughly 0.54) lies between the full conjunction and the arithmetic mean. The weight generating function associated the lowest weight (1/9th) to the fully satisfied criterion and the highest weight (5/9th, more than half) to the least satisfied criterion.

It can easily be verified that increasing the power to which x is raised in f increases the "conjunctivity" of f. The more conjunctive f is, the more weight it will assign to the worst criterion. Other examples of fuzzy quantifiers are shown in Figure 4.2.

The OWA operators have received several points of criticism. For starters, OWA operators do not discriminate between individual criteria. This means that the same weight may be distributed to a different criterion for different entities. In addition and related to this is that the weight of an input can not be set directly. To address these issues, Yager proposed an extension to OWA which adds the option to define a weight for each attribute. The resulting technique is called weighted OWA (WOWA). In WOWA, the user is required to define a weight vector $\mathbf{w}$ with as many elements as there are criteria, such that $w_i \in [0, 1]$ is the weight for criterion c_i and that

$$\sum_{i=1}^{n} w_i = 1$$



(a) Most (b) Some (c) At least half

Figure 4.2: Examples of functions that can be used to achieve linguistic "fuzzy" quantifiers with an OWA operator.

In other words, the weights should be normalized so they sum up to 1.

The procedure for computing the global suitability grade is slightly altered. When sorting the criteria according to the elementary suitability grades, the weights are sorted in the same order. Assume that the sorted weight vector is called $\mathbf{y}$. The global suitability grade is computed as:

$$s_g \overset{\text{WOWA}}{=} \sum_{i=1}^{n} [z_i x_i]$$

where the weight z_i of the ith best criterion x_i is defined as:

$$z_i = f\left(\sum_{j=1}^{i} y_j\right) - f\left(\sum_{j=1}^{i-1} y_j\right)$$

with $z_0 \triangleq 0$.

Example. Consider that evaluating three criteria c_1, c_2 and c_3 with respective weights 0.5, 0.3 and 0.2 on some arbitrary entity lead to the following respective elementary suitability grades: $(1.0, 0.2, 0.4)$. We will use the arithmetic mean $f(x) = x$ as weight-generating function to compute the global suitability grade using OWA. First, we sort the suitability grades and their associated weights according to their descending elementary suitability grade: $\mathbf{x} =< (1.0, 0.5), (0.4, 0.2), (0.2, 0.3) >$. Second, the global suitability grade is calculated:

$$\begin{aligned} s_g &= 1\,(f\,(0.5) - 0) + 0.4\,(f\,(0.5 + 0.2) - f\,(0.5)) + 0.2\,(1 - f\,(0.5 + 0.2)) \\ &= 0.5 + 0.08 + 0.06 \\ &= 0.64 \end{aligned}$$

As expected, the result 0.64 corresponds to the weighted average of the elementary suitability grades according to the specified weight vector.

Choquet/Sugeno integration

Though not invented by Choquet for the purpose of decision making [2], fuzzy integrals were adopted as aggregators for use in DSS [20, 22, 24] shortly after Sugeno independently defined a similar core concept (ordinal integration) [26]. A fuzzy integral is a computational method that is applied on a fuzzy measure defined on the powerset of all criteria. The fuzzy measure G is used to associate a weight to each possible subset of criteria. The weight of a particular set of criteria C is given by $\mathrm{G}(C)$. Hereby, the axioms of a fuzzy measure must be satisfied. For convenience, they are briefly repeated here:

$$\mathrm{G}(\emptyset) = 0$$

$$\mathrm{G}(\mathcal{U}) = 1$$

$$\forall C_1, C_2 \in \mathcal{U} : C_1 \subseteq C_2 \Rightarrow \mathrm{G}(C_1) \leq \mathrm{G}(C_2)$$

Recall that $\mathcal{U}$ denotes the universe of discourse, here representing the set containing all criteria.

Choquet's discovery of the fuzzy integral can be applied on ratio scales, whereas Sugeno's version is better suited for ordinal scales. Choquet's version of the fuzzy integral can be applied in the context of decision making as an aggregator as follows:

1. Evaluate every criterion c_i. Let thereby s_{r,c_i} represent the degree to which r satisfies c_i.

2. Sort the criteria according to these degrees from best to worst (highest to lowest value). Let $\mathbf{x}$ be this vector.

3. Add the elements of x to a solution set one by one, in order, each time raising the intermediary output (starting at 0) by the product of x_i with the increase in weight of the solution sets before and after adding this criterion. Hereby the weight of a solution set C is given by the fuzzy measure as $\mathrm{G}(C)$.

Let $O : \mathbb{N}_{\leq n} \to [0, 1]$ ($\mathbb{N}_{\leq n}$ being the n first natural numbers: $1, 2, \ldots, n$) be a weight-defining function such that $O(i) = \mathrm{G}(\bigcup_{j=1}^{i} x_j)$ corresponds to the weight of the solution set that contains the best i criteria (i.e. the first i elements of $\mathbf{x}$). Then, for the given entity, the resulting aggregated suitability s_g using the Choquet integral, computed under the fuzzy measure G whose weight-defining function is O, equals:

$$s_g \overset{FI}{=} \sum_{i=1}^{n} [x_i [O(i) - O(i-1)]]$$

By convention, $O(0) \triangleq 0$. Indeed, s_g is a weighted average of all s_{r,c_i}, and, under the assumption that all criteria and the fuzzy measure G are normalised, will always be bound between 0 and 1.

Consider a universe of two attributes, a_1 and a_2. On these attributes, two criteria c_1 and c_2 are respectively defined. A fuzzy integral to evaluate entity r will be based on a fuzzy measure of the form:

$$\begin{aligned} \mathrm{G}(\emptyset) &= 0 \\ \mathrm{G}(\mathrm{c}_1) &= w_{\mathrm{c}_1} \\ \mathrm{G}(\mathrm{c}_2) &= w_{\mathrm{c}_2} \\ \mathrm{G}(\mathrm{c}_1 \cup \mathrm{c}_2) &= 1 \end{aligned}$$

First, the criteria are evaluated and sorted. Because there are 2 criteria, there are only 2 possible scenarios:

$$\begin{aligned} s_{r,\mathrm{c}_1} \geq s_{r,\mathrm{c}_2} &\implies s_g = w_{\mathrm{c}_1} s_{r,\mathrm{c}_1} + (1 - w_{\mathrm{c}_1}) s_{r,\mathrm{c}_2} \\ s_{r,\mathrm{c}_1} < s_{r,\mathrm{c}_2} &\implies s_g = w_{\mathrm{c}_2} s_{r,\mathrm{c}_2} + (1 - w_{\mathrm{c}_2}) s_{r,\mathrm{c}_1} \end{aligned} \tag{4.8}$$

Note that if $s_{r,\mathrm{c}_1} = s_{r,\mathrm{c}_2}$, both scenarios lead to the same output, namely $s_g = s_{r,\mathrm{c}_1} = s_{r,\mathrm{c}_2}$, so it does not matter which of both scenarios includes the equality case.

In the first scenario, the behavior of the aggregator depends on w_{c_1}, but not on w_{c_2}. In the second scenario, the opposite is true. If $w_{\mathrm{c}_1} = w_{\mathrm{c}_2} = 0.5$, the aggregator calculates the average of s_{r,c_1} and s_{r,c_2} in both scenarios.

Recall that fuzzy measures are not necessarily additive, they only need to be monotonic. Herein lies the mechanism by which fuzzy integrals are able to achieve partial conjunctions and disjunctions. Consider the interpretations of super- and subadditivity for any two criteria c_1 and c_2:

- If $w_{\mathrm{c}_1} + w_{\mathrm{c}_2} > w_{\mathrm{c}_1 \cup \mathrm{c}_2}$ (superadditivity), it follows that c_1 is to a certain extent replaceable by c_2 (= partially disjunctive).
- If $w_{\mathrm{c}_1} + w_{\mathrm{c}_2} < w_{\mathrm{c}_1 \cup \mathrm{c}_2}$ (subadditivity), it follows that c_1 should be, to a certain degree, simultaneously satisfied with c_2 (= partially conjunctive).

Indeed, if c_1 and c_2 are superadditive, this means that $1 - w_{\mathrm{c}_2} < w_{\mathrm{c}_1}$ (and also $1 - w_{\mathrm{c}_1} < w_{\mathrm{c}_2}$) and weight is shifted to the best of both criteria while the other's influence on the output is lowered. Per direct consequence, it is sufficient that either has a high value for the aggregated result to have a high value too.

Dually it can be seen that subadditivity semantically represents the preference to see both c_1 and c_2 satisfied simultaneously as more impact is given to the lowest of both, meaning it will have a larger impact on the result than the other, better criterion. It follows that both must be high for the output to be high.

There is a strong similarity between fuzzy integrals and OWA. Namely, when the capacity assigns the same weight to every set with the same cardinality, the fuzzy integral results in the same outcome as OWA. Indeed, OWA is a particular case of fuzzy integration where the criteria are indistinguishable: it only matters *how many* criteria are in each intermediate solution set, not *which* criteria.

Student	Mathematics	Physics	Literature
A	18	16	10
B	10	12	18
C	14	15	15

Table 4.2: Fictive grades for students A, B and C for courses mathematics, physics and literature. The grades are given on a 0 to 20 scale.

Student	Mathematics	Physics	Literature	Weighted average
A	18	16	10	15.25
B	10	12	18	12.75
C	14	15	15	14.62

Table 4.3: Overall grades for students A, B and C computed through a simple weighted average.

Example. A textbook example for explaining the power of sub- and superadditivity, borrowed from Grabisch [20], has to do with grading students. Assume we have to compute an overall suitability grade to grade students based on their performance in three subjects: mathematics, physics and literature. Three students' grades for these courses are given in Table 4.2.

The examinator deems scientific courses more important and as such comes up with a weight of 3 for maths, 3 for physics and 2 for literature. The results of using a simple weighted average to compute the overall grade are shown in Table 4.3. One could argue that these results are not satisfying, because student C, who is equally good in all fields, should be preferrable to student A, who excels at the scientific subjects though is weak at literature. It is not possible to achieve this by changing the weights, but it is possible by using Choquet integration.

We distinguish the following properties:

1. Scientific subjects (mathematics and physics) are more important than literature.

2. Scientific subjects are somewhat related, so it is expected that generally, students that are good at either will be good at the other, too. Hence, students that perform well at both should not be favored strongly.

3. Students that are good at both science and literature are special and should be favored strongly.

These can be translated into a fuzzy measure as follows:

1. $\mathrm{G}(\{\text{maths}\}) = \mathrm{G}(\{\text{physics}\}) = 0.45$, $\mathrm{G}(\{\text{literature}\}) = 0.3$ (scientific subjects are more important).

2. $\mathrm{G}(\{\text{maths}, \text{physics}\}) = 0.5 < \mathrm{G}(\{\text{maths}\}) + \mathrm{G}(\{\text{physics}\})$ (subadditivity between mathematics and physics to indicate their inherent similarity).

Student	Mathematics	Physics	Literature	Weighted average
A	18	16	10	13.9
B	10	12	18	13.6
C	14	15	15	14.9

Table 4.4: Overall grades for students A, B and C computed through Choquet integration.

3. $\mathrm{G}(\{\text{maths}, \text{literature}\}) = \mathrm{G}(\{\text{physics}, \text{literature}\}) = 0.9 > 0.45 + 0.3$ (superadditivity to favor students good at both science and literature).

Evaluating the students with a Choquet integral using this fuzzy measure leads to the grades shown in Table 4.4. Now, the students are ordered as desired: student C is ranked highest due to his globally acceptable performance in all fields, and student B is placed behind student A due to the higher importance of scientific subjects.

4.2.4 Partial absorption operators

Partial absorption operators (PAOs) are a special case of advanced aggregation operators that are only found in GCD. They are a vague extension to the concept of regular absorption, like in $x \wedge (x \vee y) = x \vee (x \wedge y) = x$, where x absorbs y. This is achieved by replacing the conjunction and disjunction by their partial counterparts. As described by Dujmović in [14, 11], PAOs express an asymetric relation between a *dominant* and an *optional* (typically referred to as "desired") criterion. The dominant plays a decisive role in the aggregation, while the desired only serves as a modifier to the result of the dominant. The modification by the desired can be either positive (reward) or negative (penalty). The dominant criterion is said to *partially absorb* the desired criterion in the aggregation process. We have studied two types of partial absorptions: the conjunctive partial absorption and the disjunctive partial absorption.

Conjunctive Partial Absorption

The CPA operator $cpa : [0,1]^2 \to [0,1]$ models the partial absorption of a desired criterion c_d by a mandatory criterion c_m [14]. As the operator is conjunctive and c_m is mandatory (and absorbing), the output of the CPA for any entity with $s_{r,\mathrm{c}_m} = 0$ must be 0. Furthermore, low values of s_{r,c_d} must lower the output below s_{r,c_m} to penalize systems where the desired criterion is not met, whereas high s_{r,c_d} should raise the output above s_{r,c_m} as a form of reward.

Mathematically this translates into 3 requirements:

$$cpa(0, s_{r,\mathrm{c}_d}) = 0, \qquad 0 \leq s_{r,\mathrm{c}_d} < 1 \tag{4.9}$$

$$0 < cpa(s_{r,\mathrm{c}_m}, 0) < s_{r,\mathrm{c}_m}, \qquad 0 < s_{r,\mathrm{c}_m} \leq 1 \tag{4.10}$$

$$s_{r,\mathrm{c}_m} < cpa(s_{r,\mathrm{c}_m}, 1) < 1, \qquad 0 < s_{r,\mathrm{c}_m} < 1. \tag{4.11}$$

From these requirements, it is implied that $cpa(1,1) = 1$.

In order to complete the definition of the CPA, two additional parameters P and R are required, where P indicates the maximal penalty an entity can receive when $s_{r,\mathrm{c}_d} = 0$ and R indicates the maximal reward an entity can receive when $s_{r,\mathrm{c}_d} = 1$. Note that, for an entity with constant c_m, this implies there exist a value for c_d such that $0 < s_{r,\mathrm{c}_d} < 1$, where c_m fully absorbs c_d and thus $s_g = s_{r,\mathrm{c}_m}$.

There are several ways to implement a CPA operator. One option, which uses GCD operators nesting the weakest partial disjunction (the average) in a strong partial conjunction, is visualized in Figure 4.3. Here, weight parameters w_1 and w_2 can be used in combination with the strength of the partial aggregators in order to achieve the desired degree of penalty and reward. A comprehensive look-up table for translating a desired amount of penalty and reward into the connection with the weight parameters is given in [14].

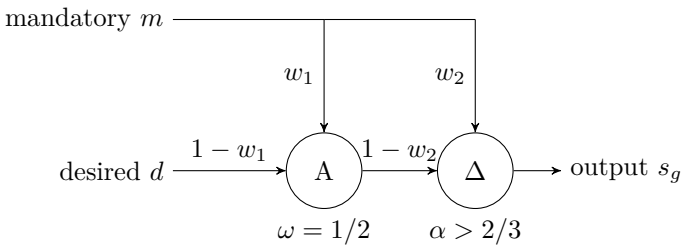


Figure 4.3: A schematic view of the CPA using GCD operators. An average is used to combine the two inputs and a strong partial conjunction to combine the intermediary output with the mandatory.

Table 4.5 shows the output of the CPA operator from Figure 4.3 with a penalty of 20% and a reward of 15% ($w_1 = 0.8$, $w_2 = 0.15$ and $\alpha = 0.875$, a strong partial conjunction such that the WPM exponent $p = -3.51$ [14]) for 121 specifically generated systems of two attributes c_m and c_d so that both s_{r,c_m} (rows) and s_{r,c_d} (columns) vary from 0 to 1 (inclusive) in steps of 0.1. The same results are presented graphically in figure 4.4, where s_{r,c_m} is displayed on the x-axis, s_g on the y-axis and each data series represents a different constant value for s_{r,c_d}.

Interestingly, the point where c_m fully absorbs c_d coincides with $s_{r,\mathrm{c}_m} = s_{r,\mathrm{c}_d}$, as indicated by the values on the table's diagonal. Note that this respects both edge cases where $s_g = s_{r,\mathrm{c}_m} = s_{r,\mathrm{c}_d} = 1$ (ideal entity) and $s_g = s_{r,\mathrm{c}_m} = s_{r,\mathrm{c}_d} = 0$ (fully unfit entity). Additionally it can be seen from the first row, $s_{r,\mathrm{c}_m} = 0$, that the CPA is indeed conjunctive and c_m is indeed mandatory.

Disjunctive Partial Absorption

The disjunctive partial absorption (DPA) operator $dpa : [0,1]^2 \to [0,1]$ implements the partial absorption of a desired criterion c_d by a sufficient criterion c_s

Table 4.5: Output of the CPA using GCD operators with $w_1 = 0.8$ and $w_2 = 0.15$.

$s_{r,c_m} \backslash s_{r,c_d}$	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
0.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.10	0.0820	0.1000	0.1158	0.1288	0.1392	0.1471	0.1530	0.1574	0.1607	0.1631	0.1649
0.20	0.1639	0.1825	0.2000	0.2164	0.2315	0.2453	0.2577	0.2687	0.2784	0.2869	0.2942
0.30	0.2459	0.2646	0.2826	0.3000	0.3166	0.3324	0.3473	0.3613	0.3744	0.3865	0.3977
0.40	0.3278	0.3466	0.3649	0.3827	0.4000	0.4167	0.4328	0.4483	0.4631	0.4772	0.4906
0.50	0.4098	0.4286	0.4470	0.4651	0.4828	0.5000	0.5168	0.5330	0.5488	0.5641	0.5788
0.60	0.4918	0.5106	0.5291	0.5474	0.5653	0.5828	0.6000	0.6168	0.6332	0.6492	0.6648
0.70	0.5737	0.5926	0.6112	0.6295	0.6476	0.6654	0.6828	0.7000	0.7168	0.7333	0.7495
0.80	0.6557	0.6745	0.6932	0.7116	0.7298	0.7478	0.7654	0.7829	0.8000	0.8169	0.8334
0.90	0.7376	0.7565	0.7752	0.7937	0.8120	0.8300	0.8479	0.8655	0.8829	0.9000	0.9169
1.00	0.8196	0.8385	0.8572	0.8757	0.8941	0.9123	0.9302	0.9480	0.9655	0.9829	1.0000

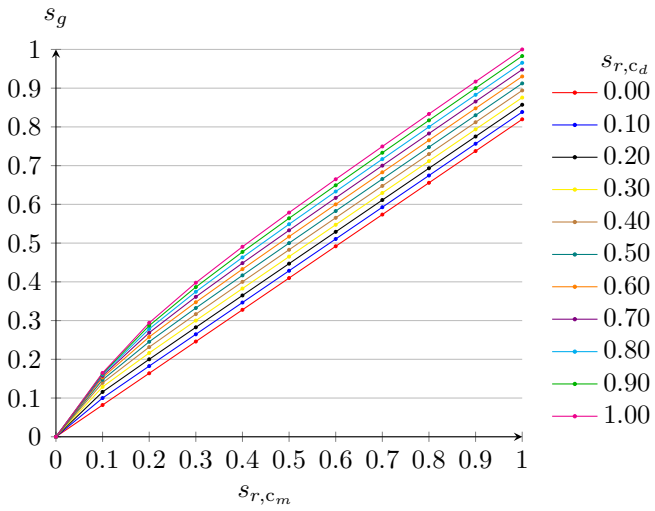


Figure 4.4: Output of the CPA using GCD operators.

[14]. If $s_{r,c_s} = 1$, s_g should be 1 as c_s is considered sufficient reason to accept r, regardless of c_d. Otherwise, if $s_{r,c_s} < 1$, c_d should (similar to the case of the CPA) modify the output by a prespecified penalty or reward depending on s_{r,c_d}.

Formally, this is summarized by the following constraints:

$$dpa(1, s_{r,c_d}) = 1, \qquad 0 \leq s_{r,c_d} \leq 1 \tag{4.12}$$

$$0 < dpa(s_{r,c_s}, 0) < s_{r,c_s}, \qquad 0 < s_{r,c_s} < 1 \tag{4.13}$$

$$s_{r,c_s} < dpa(s_{r,c_s}, 1) < 1, \qquad 0 \leq s_{r,c_s} < 1. \tag{4.14}$$

Similarly to the CPA, the DPA also has the concepts of penalty and reward. A way of implementing the DPA with GCD operators by nesting a weak partial conjunction with a disjunction is visualized in Figure 4.5.

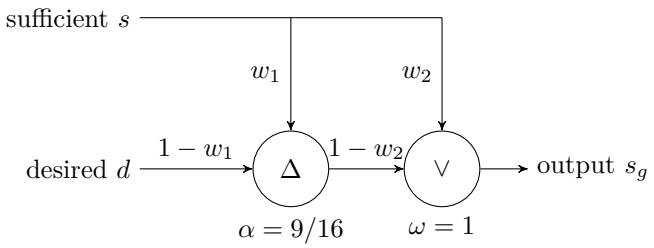


Figure 4.5: A schematic view of the DPA using GCD operators. An average is used to combine the two inputs and a strong partial conjunction to combine the intermediary output with the sufficient.

Table 4.6: Output of the DPA using GCD operators with $w_1 = 0.8$ and $w_2 = 0.15$.

$s_{r,c_s} \backslash s_{r,c_d}$	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
0.00	0.0000	0.0143	0.0286	0.0429	0.0572	0.0715	0.0858	0.1001	0.1144	0.1287	0.1430
0.10	0.1000	0.1000	0.1272	0.1513	0.1737	0.1951	0.2157	0.2358	0.2554	0.2746	0.2936
0.20	0.2000	0.2000	0.2000	0.2284	0.2545	0.2791	0.3026	0.3254	0.3475	0.3691	0.3902
0.30	0.3000	0.3000	0.3000	0.3000	0.3288	0.3559	0.3817	0.4065	0.4305	0.4539	0.4768
0.40	0.4000	0.4000	0.4000	0.4000	0.4000	0.4291	0.4567	0.4833	0.5089	0.5338	0.5581
0.50	0.5000	0.5000	0.5000	0.5000	0.5000	0.5000	0.5293	0.5573	0.5843	0.6106	0.6361
0.60	0.6000	0.6000	0.6000	0.6000	0.6000	0.6000	0.6000	0.6294	0.6577	0.6851	0.7118
0.70	0.7000	0.7000	0.7000	0.7000	0.7000	0.7000	0.7000	0.7000	0.7295	0.7580	0.7857
0.80	0.8000	0.8000	0.8000	0.8000	0.8000	0.8000	0.8000	0.8000	0.8000	0.8295	0.8582
0.90	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000	0.9296
1.00	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 4.6 shows the output of the aggregator applied on the same 121 combinations of inputs between the sufficient and desired inputs. The same results are plotted in Figure 4.6 (using weighted power means with $w_1 = 0.8$, $w_2 = 0.15$ and $\alpha = 9/16$, a weak partial conjunction resulting in exponent $p = 0.619$). It is clear that the sufficient condition is respected (i.e. if $s_{r,c_s} = 1$, $s_g = 1$). Like with the CPA, the (average) degrees of penalty and reward (shaping the curve) can be influenced by manipulating the weights and the stengths of the (partial) aggregators.

4.3 Conjunctive Partial Absorption using Fuzzy Integrals

4.3.1 Challenge

Fuzzy measures are capable of modelling mandatory requirements [2, 24, 22]. To express that it is mandatory that condition c_m is satisfied, one must construct the fuzzy measure such that every set that does not contain c_m is assigned a

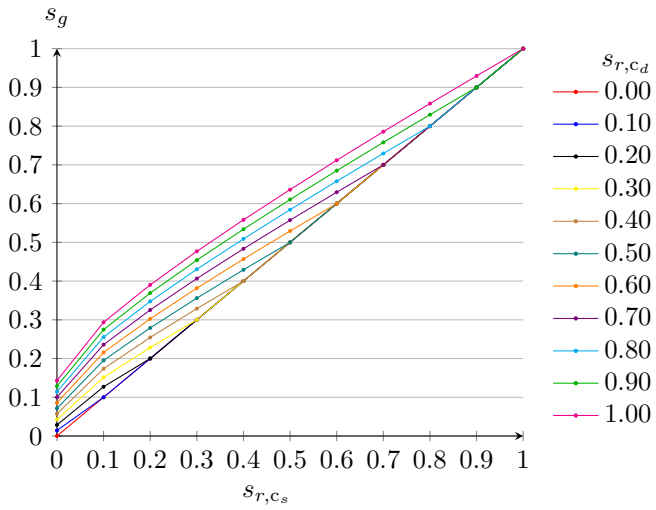


Figure 4.6: Output of the DPA using GCD operators.

weight of 0. This is required because otherwise it is not guaranteed that the score for a system r is 0 if $s_{r,c_m} = 0$. Indeed, it can be seen from Eq. 4.8 that when $s_{r,c_m} = 0$ and $s_{r,c_d} > s_{r,c_m}$, $s_g = 0$ only if $w_d = 0$.

If $w_d = 0$, we have either:

$$s_g = \begin{cases} w_m s_{r,c_m} + (1 - w_m) s_{r,c_d} & s_{r,c_d} < s_{r,c_m} \\ 0 s_{r,c_d} + (1 - 0) s_{r,c_m} & s_{r,c_d} \geq s_{r,c_m} \end{cases}$$

It should be noted that in the second case, c_d has no impact on the output, which implies that whenever $s_{r,c_d} > s_{r,c_m}$, the result relies solely on c_m. Consequently, this fuzzy measure does not correspond with the formal definition of the CPA. For c_d to have an impact, w_d must be larger than 0. This, however, conflicts with the demand that w_d must be 0 for c_m to be mandatory.

In this section, a strategy to implement the CPA operator using fuzzy intregration is proposed. An earlier version of the contents of this section were presented at FUZZ-IEEE 2016 and subsequently published in the proceedings of the conference.

4.3.2 Solution

To ensure that $s_g = 0$ if $s_{r,c_m} = 0$ (wherein always $s_{r,c_m} < s_{r,c_d}$ and thus $s_g = w_d s_{r,c_d}$), it is absolutely necessary that $w_d = 0$. If $s_{r,c_m} > 0$, however, it is not necessarily required that s_g must be 0, nor does it follow that its implication (that w_d must be 0) still holds. Hence, to reconcile the constraint that c_m is mandatory with the requirement that c_d should have an impact on the output if $s_{r,c_m} > 0$, we must redefine w_d to be non-zero when c_m is satisfied. As such, we propose to redefine w_d as a function that depends on c_m, as follows:

$$w_d = \begin{cases} 0 & s_{r,\mathrm{c}_m} = 0 \\ r \in \,]0,1] & s_{r,\mathrm{c}_m} > 0 \end{cases}$$

Hereby it is especially important to carefully study the range of this function so that the monotonicity of the fuzzy measure is not violated. Because c_d is a singleton, w_d must be bounded by the weight of the (maximal) 0-set and the minimal 2-set containing c_d. For the CPA, where there are just two inputs, c_m and c_d, there is only one 0-set (which is the empty set and which always has weight 0) and exactly one 2-set (which represents the universe and hence has weight 1). This implies that the range of w_d is equal to $[0, 1]$, which was already automatically implied by the normalization constraint.

Recall the example given in 4.2.4. Table 4.7 shows the output of a fuzzy integral defined this way when applied on the same 121 systems. Here, $w_m = 0.8$ and for the definition of w_d, r was chosen to be 0.15. The same results are shown graphically in figures 4.7 (2D) and 4.8 (3D).

Table 4.7: Output of the CPA (P = 20%, R = 15%) using a Fuzzy Measure

	s_{r,c_d}										
s_{r,c_m}	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
0.00	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
0.10	0.08000	0.10000	0.11500	0.13000	0.14500	0.16000	0.17500	0.19000	0.20500	0.22000	0.23500
0.20	0.16000	0.18000	0.20000	0.21500	0.23000	0.24500	0.26000	0.27500	0.29000	0.30500	0.32000
0.30	0.24000	0.26000	0.28000	0.30000	0.31500	0.33000	0.34500	0.36000	0.37500	0.39000	0.40500
0.40	0.32000	0.34000	0.36000	0.38000	0.40000	0.41500	0.43000	0.44500	0.46000	0.47500	0.49000
0.50	0.40000	0.42000	0.44000	0.46000	0.48000	0.50000	0.51500	0.53000	0.54500	0.56000	0.57500
0.60	0.48000	0.50000	0.52000	0.54000	0.56000	0.58000	0.60000	0.61500	0.63000	0.64500	0.66000
0.70	0.56000	0.58000	0.60000	0.62000	0.64000	0.66000	0.68000	0.70000	0.71500	0.73000	0.74500
0.80	0.64000	0.66000	0.68000	0.70000	0.72000	0.74000	0.76000	0.78000	0.80000	0.81500	0.83000
0.90	0.72000	0.74000	0.76000	0.78000	0.80000	0.82000	0.84000	0.86000	0.88000	0.90000	0.91500
1.00	0.80000	0.82000	0.84000	0.86000	0.88000	0.90000	0.92000	0.94000	0.96000	0.98000	1.00000

4.3.3 Interpretation

Let us briefly analyze our approach. When $s_{r,\mathrm{c}_m} = s_{r,\mathrm{c}_d}$, there should be no penalty nor reward. This requirement is automatically satisfied, as it is a property of fuzzy integrals that when all inputs are equal, the output is equal to the inputs, too. If, for some entity, the desired criterion is more satisfied than the mandatory requirement ($s_{r,\mathrm{c}_d} > s_{r,\mathrm{c}_m}$), the output should be boosted so as to reward that entity:

$$s_g = w_d s_{r,\mathrm{c}_d} + (1 - w_d) s_{r,\mathrm{c}_m}$$

Indeed, the overall s_g contains a reward term $w_d s_{r,\mathrm{c}_d}$.

The amount of reward is maximal (equal to w_d) when $s_{r,\mathrm{c}_d} = 1$. In this case, $s_g = w_d + (1 - w_d) s_{r,\mathrm{c}_m}$. Note that in the special case that $s_{r,\mathrm{c}_m} = 0$, $w_d = 0$ so that $s_g = 0$. In all other cases, $w_d = r$ and it is clear from the definition that s_g can be directly influenced by choosing a value for r. It is easy to see

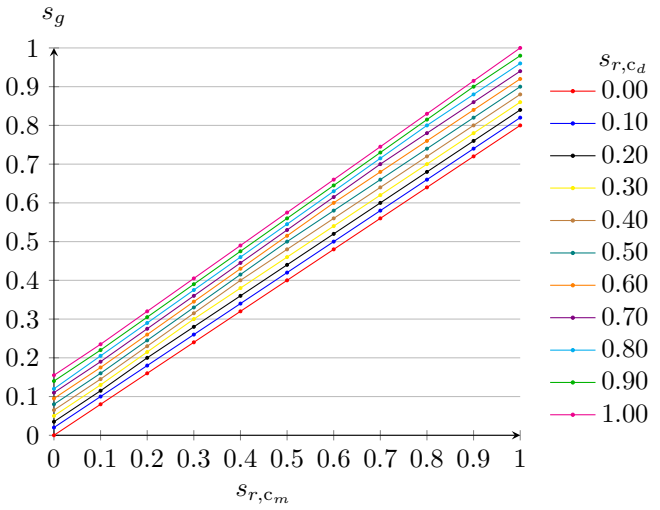


Figure 4.7: Results of the fuzzy measure approach in 2D.

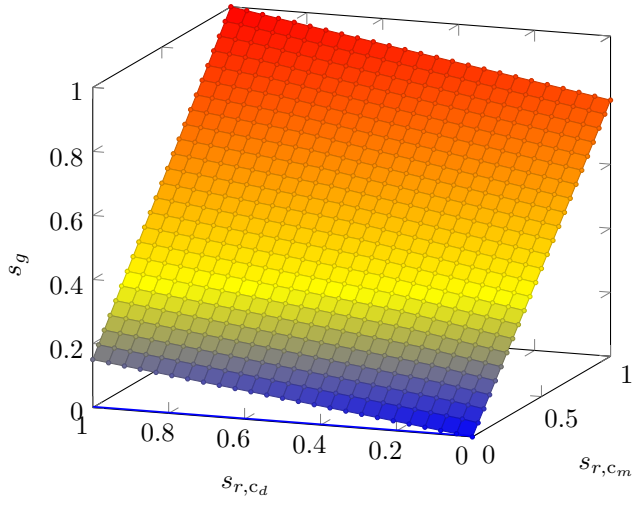


Figure 4.8: Results of the fuzzy measure approach in 3D.

that $s_g \sim r$ (increasing r increases s_g), so increasing r effectively increases the maximal reward that is given by the operator, regardless of c_m and w_m.

Dually, if the desired criterion is only weakly satisfied compared to the mandatory requirement ($s_{r,\mathrm{c}_m} \geq s_{r,\mathrm{c}_d}$), c_d should lower s_g to reflect that the operator assigns a penalty to the entity under evaluation. When $s_{r,\mathrm{c}_d} = 0$, this penalty should be maximal. In that case, $s_g = w_m s_{r,\mathrm{c}_m}$. As such, it follows that s_g is bounded from below by $w_m s_{r,\mathrm{c}_m}$. When $s_{r,\mathrm{c}_m} = 1$, $s_g = w_m$, and the maximal penalty is $1 - w_m$. Clearly, w_m acts as a modifier that can be used to define the maximal amount of penalty an entity can receive.

In summary, the *maximal penalty* equals $1 - w_m$ and the *maximal reward* equals r. Note that these semantics override the regular interpretations of sub- and superadditivity. If one would want to award a maximal reward of 20% and a maximal penalty of 10%, one would set $r = 0.2$ and $w_m = 0.9$.

4.3.4 Reflection

Visually, the output generated by the proposed solution very closely resembles that of Dujmović's original solution based on the GCD operator, but is different in the region approaching (0, 0). From Figure 4.8 it can be seen that there is a discontinuity in the suitability surface along the s_{r,c_d}-axis when s_{r,c_m} approaches 0. Indeed, s_g should be 0 for *all* values of s_{r,c_m}, regardless of the value of s_{r,c_d}, whereas now, it is only 0 when $s_{r,\mathrm{c}_d} = 1$. This is not in line with the results of the original implementation using GCD operators, and it is undesirable because it implies that even a tiny change in c_m near $s_{r,\mathrm{c}_m} \approx 0$ has a non-negligable impact on s_g. This is especially troubling if the measuring of c_m might be (and, in practice, most likely will be) subject to noise or some other form of uncertainty. Such undesirable "un-fuzzy" behavior is to be avoided.

In order to rectify this, w_d must be continuous rather than piecewise continuous and still produce 0 when $s_{r,\mathrm{c}_m} = 0$. Any function of the form $w_d = \gamma s_{r,\mathrm{c}_m}$ can be used as long as γ does not cause w_d to exceed its allowed range.

As per illustration, consider the following, alternative definition of w_d:

$$w_d = r s_{r,\mathrm{c}_m} \tag{4.15}$$

This definition of w_d is intentionally continuous. Moreover, $w_d = 0$ when $s_{r,\mathrm{c}_m} = 0$. The results of applying this function on the same 121 entities is plotted in figures 4.9 (2D) and 4.10 (3D). Where $s_{r,\mathrm{c}_d} = 1$ (top-most pink trend line), the output is parabolic and fills the entire spectrum between 0 and 1. Where $s_{r,\mathrm{c}_d} = 0$ (bottom-most red trend line), the aggregated score is linear with c_m and confined between 0 and $1 - w_m$ (maximal penalty). These results match the geometric interpretation of reward and penalty as described in [14]. As intended, these modifications have made the aggregator resilient to minor fluctuations of c_m near 0.

However, note that the maximal reward is no longer equal to r. Let us try to compute the maximal reward. In any case, we know that a reward is only assigned if $s_{r,\mathrm{c}_d} > s_{r,\mathrm{c}_m}$. Recall that this means that the output is computed as

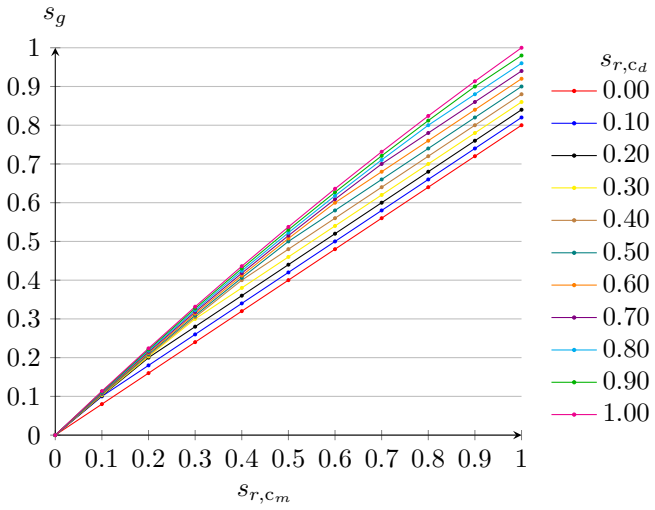


Figure 4.9: Results using continuous definition of w_d in 2D.

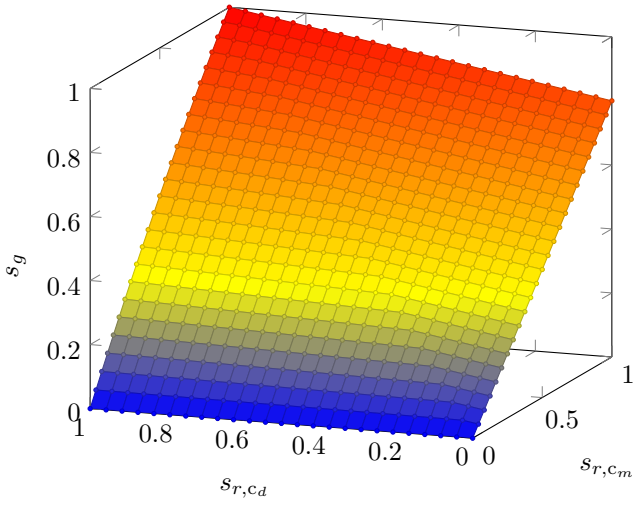


Figure 4.10: Results using continuous definition of w_d in 3D.

follows:

$$s_g = rs_{r,\mathrm{c}_m} s_{r,\mathrm{c}_d} + (1 - rs_{r,\mathrm{c}_m}) s_{r,\mathrm{c}_m} \tag{4.16}$$

Obviously, the better c_d is satisfied, the more reward is given. The maximal reward is obtained when $s_{r,\mathrm{c}_d} = 1$. In that case, the reward term becomes equal to rs_{r,c_m}.

If there were no desired criterion, the overall suitability grade s_g would be exactly equal to the degree to which the (only remaining) mandatory criterion is satisfied, i.e. s_{r,c_m}. Hence, the actual amount of reward, r^*, obtained by adding the partial absorption of the desired criterion, is given by the difference between eq. 4.16 (with $s_{r,\mathrm{c}_d} = 1$) and s_{r,c_m}:

$$\begin{aligned} r^* &= rs_{r,\mathrm{c}_m} + (1 - rs_{r,\mathrm{c}_m}) s_{r,\mathrm{c}_m} - s_{r,\mathrm{c}_m} \\ &= rs_{r,\mathrm{c}_m}(1 - s_{r,\mathrm{c}_m}) \end{aligned}$$

In order to derive the maximal amount of reward from this equation, in function of s_{r,c_m}, we must differentiate it to s_{r,c_m} and solve it for s_{r,c_m}. In doing so, we find that r^* is maximal for $s_{r,\mathrm{c}_m} = \frac{1}{2}$. At this point, the maximal amount of reward is found to be $r^* = \frac{r}{4}$. In other words, in order to again obtain a maximal reward of 0.2, one would have to set r to 0.8. Indeed, one would then find:

$$\begin{aligned} s_g &= \frac{8}{10} s_{r,\mathrm{c}_m} + \left(1 - \frac{8}{10} s_{r,\mathrm{c}_m}\right) s_{r,\mathrm{c}_m} \\ &= \frac{s_{r,\mathrm{c}_m}(9 - 4s_{r,\mathrm{c}_m})}{5} \end{aligned}$$

For $s_{r,\mathrm{c}_m} = \frac{1}{2}$, this yields $\frac{7}{10}$, which is indeed 0.2 higher than 0.5.

One could use other reward functions as long as they are continuous and 0 for $s_{r,\mathrm{c}_m} = 0$, but then one would also have to re-examine the impact on the maximal reward and adjust the parameters of the reward function accordingly in order to achieve to desired behavior. Note that this discussion is irrelevant for maximal penalty because there is no discontinuity for $s_g = 1$. If absolutely required, one could, however, also replace the maximal penalty by a function in order to obtain symmetry between penalty and reward.

4.3.5 Usability in hierarchical compositions

The largest downside of fuzzy integrals is the complexity of defining the fuzzy measure, as it is exponential in the amount of criteria. To overcome this, several techniques such as the Sugeno λ-measure [29] and k-additivity [21] have been proposed. Additionally, Sugeno studied the possibility of hierarchically decomposing a large set of criteria into smaller sets of strictly interdependent criteria where possible to further reduce complexity by avoiding combinations of criteria that make no sense from a semantic point of view [27, 25]. In line with this, it is mentioned here that the proposed method for the CPA can be used as an independent aggregation operator as part of a larger, hierarchical network of aggregators.

Table 4.8: 4 Fictive Hotels Near the Conference Site

Name	**Distance** (km)	**Price** (euro)
Hotel A	0.6	178
Hotel B	0.2	57
Hotel C	0.8	123
Hotel D	0.4	123

4.3.6 Example

Assume you are looking for a hotel to stay in for the duration of a conference. For this simple example, only consider two selection criteria:

- c1 = price must be below threshold (mandatory)
- c2 = preferably in proximity to the conference venue (desired)

Additionally, to reflect your preference regarding proximity is not linearly correlated to the actual distance, you would like to assign a larger penalty to those hotels further away than you would like to reward those close by. Furthermore, assuming all hotels are of equal quality, a very attractive price might persuade you to walk a longer distance. As such, you choose to evaluate a set of hotels using a CPA operator with a reward of 10% and a maximal penalty of 25%. Using the proposed technique, you need only to assign $r = 0.1$ and $w_m = 1 - 0.25 = 0.75$ to model your preferences.

Next we translate the criteria mathematically using membership functions. For the first criterion, ideally the price per night should be between 80 and 120 euro. We feel this price is acceptable and is probably reliable as a reference for a certain quality of service. Additionally, anything above 150 euro is considered unacceptable. Furthermore, the lower the price below 80 euro, the more suspicious (and thus less preferable) we deem the hotel. As such, the following criterion expresses the price is hard-bound from above by 150 euro, though that a certain quality is expected, corresponding to the ideal price range between 80 and 120 euro.

For the second criterion, consider we are only interested in those hotels within a 1km range of the conference venue. The closer to the conference, the better.

These criteria are displayed graphically as membership functions in figures 4.11 and 4.12.

Finally, the operator is used to evaluate 4 fictive hotels, displayed in table 4.8, in order to score and rank them from best to worst. For simplicity, we consider that the properties of the hotels are well-known.

From table 4.9 follows that the computed ranking from best to worst is D - C - B - A. Analyzing these results, it is no surprise that A finishes last as its price is too high. As this violates the mandatory criterion, its final score is 0 to indicate that this hotel should be fully rejected.

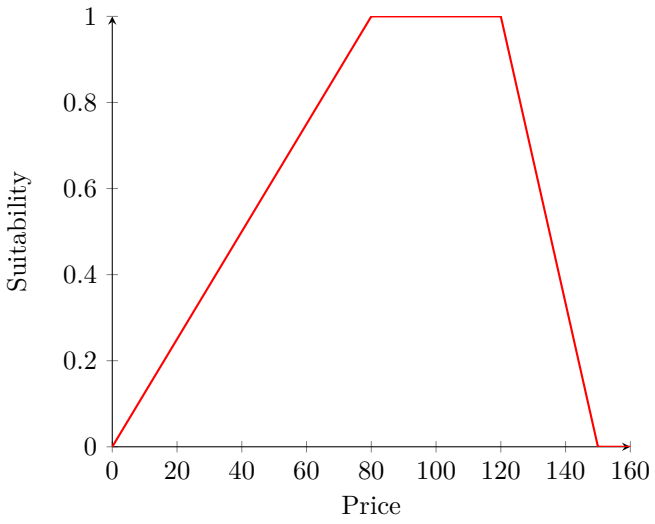


Figure 4.11: Mandatory criterion represented by a membership function reflecting preference regarding price.

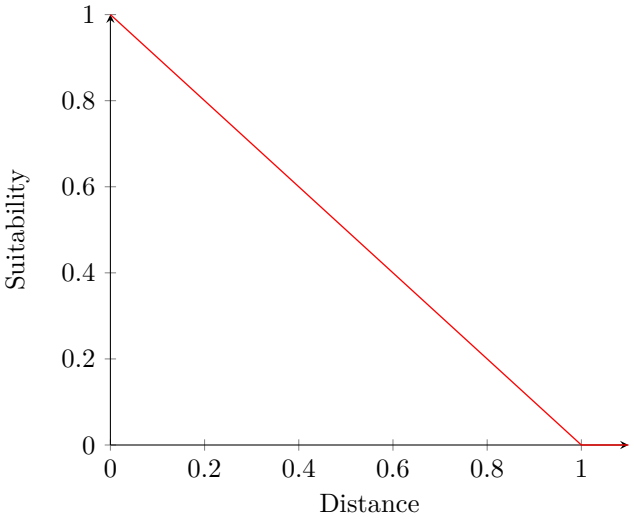


Figure 4.12: Desired criterion represented by a membership function reflecting preference regarding distance.

Table 4.9: Evaluation Results

Name	c1(distance)	c2(price)	Score
Hotel A	0.4000	0.0000	0.00000
Hotel B	0.8000	0.7125	0.72125
Hotel C	0.2000	0.9000	0.72500
Hotel D	0.6000	0.9000	0.82500

Furthermore, hotels C and D have the same price, but as hotel D is closer to the venue, it is attributed a higher reward and thus ranked before C.

Last, hotel B is in very close competition with hotel C. Despite its very close proximity (which actually results in a slight bonus for hotel B), the (perhaps suspiciously) low price has the largest impact on the overall score. However, its final score is comparable to that of hotel C because the reward given by the close proximity compensates for the strong penalty given to hotel C due to its relatively large distance.

In any case, hotel D is the clear winner as it satisfies both requirements to a good extent simultaneously, as was expected.

4.4 Aggregating suitability distributions

In this section, we will examine how aggregation operators can be applied on suitability distributions.

The aggregation operators introduced above were designed to aggregate grades of suitability. Suitability distributions, being uncertain grades of suitability, fall into this broad category but, obviously, aggregation operators must be extended in order to remain applicable. In order to aggregate suitability distributions, we consider two options: defuzzifying the distributions and aggregating the thus obtained suitability grades using unmodified, existing techniques, or aggregating the distributions directly into a single, global suitability distribution. In the following, we will discuss both approaches and motivate how to choose between them. Some of the proposed defuzzification techniques are novel contributions, as they are tightly coupled to the concept of suitability distributions, while direct aggregation is mostly based on prior research on aggregating fuzzy numbers. However, its application in a suitability context is also new.

4.4.1 Defuzzification before aggregation

Recall that the defuzzification of a suitability distribution onto a suitability grade through the strategies introduced in section 3.4 corresponds to expressing a specific tolerance, or attitude, towards uncertainty. Besides the obvious advantage that defuzzifying suitability distributions before aggregation makes it possible to use the existing aggregation operators without needing to modify them, it is also worth to consider this approach if there is a desire to express a different attitude towards the uncertainty on different criteria. This can be

especially useful in a decision support context, where certain attributes may be more critical than others and where defuzzification would usually have to be performed sooner or later anyway in order to arrive at the end-goal of obtaining a single grade per entity so these can be used to rank the entities. It can be shown that any result that can be produced by defuzzifying first should be a specific case that could otherwise be obtained by aggregating first and defuzzifying second. Indeed, a combination of all possible values can not become impossible. Nonetheless, it is yet to be investigated whether or not each particular outcome that can be obtained by choosing arbitrary defuzzification techniques before aggregating is also reachable using one of the mentioned defuzzification strategies with known semantics.

In any case, opting to defuzzify first comes with a general loss of information. The fact that the resulting aggregated global suitability grade has no remaining indicators of uncertainty indicates this. By naively defuzzifying before aggregating, it is no longer possible to distinguish an entity with uncertainty from one without.

In section 3.4, we also mentioned ways to compare suitability distributions after defuzzification based on not only a suitability grade but also on a quantification of uncertainty that can be derived from the elementary suitability distribution. We have already discussed some of the properties of such a quantification of uncertainty, such as that it is bound between 0 (fully certain) and 1 (fully uncertain). In an attempt to not lose all indicators of uncertainty, we will explore if we can aggregate defuzzified suitability-uncertainty quantification representations. The outline of the approach is that we aggregate the defuzzified suitability grades first, and then independently also the uncertainty grades in a similar fashion. However, it will quickly become clear that the aggregation of uncertainty grades can not be done completely independently. What follows is a summary of the results found by studying this subject. These findings were presented at IFSA 2015 and published in the proceedings of the conference [3].

Aggregating grades of uncertainty

In this section, we propose a novel strategy to incorporate uncertainty in the aggregation process. We will do this by extending GCD aggregators. Traditionally, a GCD aggregator is defined as a function of the form $F : [0,1]^n \to [0,1]$. Our extension, Extended GCD (EGCD), is defined as a vector function $\overline{F} : ([0,1],[0,1])^n \to ([0,1],[0,1])$, where the abscissa of an input maps to the original suitability grade of GCD and the ordinate corresponds to a grade of uncertainty. During this research, we pay close attention to the fact that the global suitability calculations are not influenced. Hence, in the case that there is no uncertainty on the inputs, the proposed technique is identical to regular GCD.

Expected behaviour As *elementary uncertainty* reflects how much doubt exists regarding the value of its corresponding input's elementary suitability, it

follows that the aggregated *global uncertainty* expresses doubt on the system's global suitability.

To guard that uncertainty can not be lost or created through aggregation we define the concept of *preservation of uncertainty*. Mathematically this translates to the property that the global uncertainty can not drop below the lowest elementary uncertainty nor can it rise above the highest elementary uncertainty. This is also true for aggregated suitability and is called "internality"[17, 9].

Regarding the calculation of the global uncertainty grade, it is easy to see that this grade should not only depend on the initial elementary uncertainty grades of the inputs, but also on their corresponding elementary suitability grades and the way these are aggregated. Let us clarify why through an example. Consider two elementary suitability-uncertainty couples: $(0.3, 0.9)$ and $(0.8, 0.4)$. Though it makes sense that the logical conjunction of the elementary suitability grades of these couples is equal to 0.3 (the lowest of the two suitability grades), it would not make sense to say that the aggregated uncertainty equals 0.4 (which one would find by also taking the lowest of the two uncertainty grades), because that would imply that the conjunction did not select the uncertainty that is associated to the worst suitability (as it should), but instead chose the uncertainty of the other input. One should expect that the result of the conjunction would be equal to $(0.3, 0.9)$.

In what follows, we first study the conjunction and disjunction as they denote the edge cases when it comes to aggregation. Afterwards, we take a look at the other aggregators.

On strict aggregators Considering the full, pure disjunction and full, pure conjunction as aggregators that *select* a certain input, we expect them to propagate that input's uncertainty to the output. As such, a strict aggregator's output should yield its *dominant* (i.e., highest for disjunction and lowest for conjunction) input, for both suitability and uncertainty.

However, if multiple (or all) inputs have an equal elementary suitability grade, then the selection of the dominant input is no longer trivial. In that case, we must first determine the set of inputs that are equally dominant (all inputs with the same elementary suitability) and then apply a tiebreaker to select the most dominant one. In case of a disjunction, this set would contain all inputs with the highest elementary suitability grade. Dually, in case of a conjunction, the set would contain all inputs with the lowest elementary suitability grade. To select the most dominant input from that set with respect to the aggregator, we rely on the elementary uncertainty grades as tiebreaker. In case of a disjunction, we are free to choose the "best" input, so we prefer the one with the lowest uncertainty. Oppositely, in case of a conjunction, we must select the "worst" input, and so we take the one with the highest uncertainty. The aggregation logic for breaking ties follows the inverse aggregation rules of the aggregator (lowest uncertainty for disjunction, highest uncertainty for conjunction). Otherwise, it could be seen as a lexicographical ordering on the suitability-uncertainty couples, but by descending order for suitability and

ascending order for uncertainty. Note that if we would first translate the indicators of uncertainty to indicators of certainty (e.g. by taking the complement to 1), the regular aggregation rules would apply. However, to keep in line with all of the above, we will keep using indicators of uncertainty.

On general aggregators Except for the strict ones, aggregators generally do not simply select a single input but rather produce a weighted combination of all inputs. Thus, in the general case, we propose to compute the global uncertainty grade as a weighted combination of all elementary uncertainty grades, given a set of suitability-uncertainty input couples. The idea is to use a weighted mean to do this, but as we already showed in section 4.4.1, using the user-defined weights associated to the inputs would lead to incorrect results. Rather than using the user-defined weights, we instead propose that the importance of each input is determined by how much "impact" that input has on the outcome of the suitability aggregation. As such, we define the *dominance* of an input, in the context of a given aggregator, as the difference between the aggregated suitability grade and that input's elementary suitability grade. If an input's elementary suitability is similar to the global suitability (both high or both low), we say that the aggregator *prefers* this input and we assign a high weight to that input's uncertainty grade. Dually, if the input and the output are dissimilar, we say the aggregator does not prefer that input and consequently assign a low weight to that input's elementary uncertainty grade. This implies that for a partial disjunction, the elementary uncertainties of inputs with high elementary suitability grades will be preferred, whereas for a partial conjunction, the elementary uncertainties of inputs with low elementary suitability grades will receive higher weights.

It is important that we validate that this approach does not violate the special cases that comprise the strict aggregators. Furthermore, we want the output of the aggregation operator to form a smooth gradient between the edge cases, to avoid that a differential in one of the inputs has a large effect on the output.

For binary aggregators (with exactly two inputs), we propose using a simple weighted average. The weight for each input is based on its dominance, which is normalised based on the dominances of both inputs. Some results are summarized in Table 4.10. This table shows the output of the 17 degrees of GCD operators (rows) applied on several combinations of suitability-uncertainty input couples (columns). One can see that the trend of the global uncertainty meets our expectations: in the case of a full conjunction, the uncertainty of the input with lowest suitability is selected, whereas in the case of a full disjunction, the uncertainty of the input with highest suitability is propagated. For all other cases (i.e. means), the aggregated uncertainty is a weighted combination of the elementary uncertainties. We also observe that the uncertainty calculations respect the mandatory behaviour of the forms of the hard partial conjunction.

Aggregators that work on more than two inputs, such as compound aggregators [4, 18], have not yet been investigated.

r	(1.0; 1.0), (0.0; 0.0)	(1.0; 0.0), (0.0; 1.0)	(0.2; 0.75), (0.7; 0.5)
C	0.000; 0.000	0.000; 1.000	0.200; 0.750
C++	0.000; 0.000	0.000; 1.000	0.216; 0.742
C+	0.000; 0.000	0.000; 1.000	0.243; 0.729
C+−	0.000; 0.000	0.000; 1.000	0.283; 0.708
CA	0.000; 0.000	0.000; 1.000	0.326; 0.687
C−+	0.000; 0.000	0.000; 1.000	0.363; 0.668
C−	0.070; 0.070	0.070; 0.930	0.394; 0.653
C−−	0.326; 0.326	0.326; 0.674	0.421; 0.639
A	0.500; 0.500	0.500; 0.500	0.450; 0.625
D−−	0.620; 0.620	0.620; 0.380	0.481; 0.609
D−	0.709; 0.709	0.709; 0.291	0.516; 0.592
D−+	0.780; 0.780	0.780; 0.220	0.552; 0.574
DD	0.838; 0.838	0.838; 0.161	0.588; 0.556
D+−	0.887; 0.887	0.887; 0.112	0.621; 0.539
D+	0.930; 0.930	0.930; 0.070	0.651; 0.525
D++	0.967; 0.967	0.967; 0.033	0.677; 0.512
D	1.000; 1.000	1.000; 0.000	0.700; 0.500

Table 4.10: The output of the EGCD function for several binary configurations of inputs.

Example

We will demonstrate the proposed approach with an example. Imagine going on a holiday to another country. There are multiple travel options, such as going by car (system A), by train and public transportation in general (system B) or by airplane (system C). We consider only two attributes for these travel options:

- estimated travel time to reach the destination, and
- perceived comfort of the transportation method.

There are many other attributes but for the example to remain easy to follow, we limit ourselves to these two.

Assume the following about the travel options.

The car system shows slight uncertainty on travel time which stems from the possibility of running into traffic jams or being rerouted due to road works. On the other hand, the comfort of travelling by car is rated as “high” as it offers a high level of freedom: it is not only possible to go exactly where you want when you want, but you can pack a lot of luggage and keep it close to yourself, safe and comfortable.

The train (and public transportation in general) system has a longer travel time compared to the car system. Despite its higher cruise speed, the train does not go door-to-door, often resulting in multiple transfers during transit, which is especially true for longer voyages. In combination with this, public transportation is scheduled according to time tables, and transferring typically comes with added waiting time. There is also a wider degree of uncertainty

System	Attribute	Suit.	Uncert.
Car	Travel time	0.833	0.2
	Comfort	1.000	0.0
Train	Travel time	0.333	0.3
	Comfort	0.300	0.1
Airplane	Travel time	1.000	0.0
	Comfort	0.700	0.2

Table 4.11: Computed elementary suitability and uncertainty grades for the three systems.

regarding travel time, as public transportation is victim to many circumstances, mostly out of your own control, that might introduce delays. The comfort level of the train system is considered to be "low", as the hassle of carrying your luggage around from one transport to the next and the fact that you are limited in what you can take with you are both restricting factors to your freedom. On top of this, public transportation rarely comes with privacy, which some value highly.

In comparison to the others, airplane system is the fastest option. For this attribute, we consider no uncertainty as airplanes are usually very punctual (though this is certainly debatable, it serves more to illustrate a point in the calculations that follow). The airplane system is rated between the car and the train system for its comfort level as personal freedom is still restricted, however a lot of work is done for you during departure, arrival and the flight itself (mostly regarding luggage). There is a certain degree of uncertainty on this value, however, as it does happen from time to time that luggage is lost or your seat in the airplane is unfortunately close to a source of nuisance.

Before we can begin evaluating these systems, we need to define our preferences in the form of an evaluation criterion. This criterion is a combination (or superposition thereof) of elementary criteria on the attributes. Such an elementary criterion has to accurately reflect our preferences with respect to acceptable and unacceptable values from the domain of the corresponding attribute. Because being acceptable or not is considered to be a matter of degree, a regular fuzzy set that is defined over the set of valid domain values can be used to represent the user preferences.

We identify the following set of elementary criteria:

- C1: Travel time is preferably short.
- C2: Comfort should be as high as possible.

Assume that evaluating the systems with concrete implementations of these criteria leads to elementary suitability distributions which are defuzzified using an optimistic strategy (assuming normal travel conditions). The results thereof are shown in Table 4.11.

In the next step, the elementary suitability grades, with their corresponding elementary uncertainty grades, are aggregated to a global suitability and global

System	Suitability	Uncertainty
Car	0.914	0.103
Train	0.316	0.198
Airplane	0.840	0.107

Table 4.12: Computed global suitability and uncertainty scores for three sample systems.

uncertainty for each system. To combine these criteria, we use a linguistic technique proposed by Dujmović [10]. Therefore, we first need to determine if these requirements are replaceable or not. As we want them to be true simultaneously, they are not replaceable and we will compute the degree of andness to find the correct degree of partial conjunction to combine them. Then, we add a linguistic level of importance to each requirement. We say the comfort requirement is of *high importance* whereas the travel time requirement is only considered to be of *medium importance*. As such, we find the aggregator should be a *Weak Conjunction* [28].

The resulting outcomes for the three alternatives are shown in Table 4.12. Based on these results, we decide that the train is the worst option as its global suitability is very poor compared to the other systems. Additionally, its global uncertainty grade is the highest of all. Given we entered expected values for the attributes and the interpretation of uncertainty is a margin for worse cases, this implies the train system's satisfaction might be more disappointing than indicated by its global suitability indicator. As such, we reject the train system. The car system scores better than the airplane system in both suitability and uncertainty, and is hence likely the best option, though both are viable alternatives.

Interpretation

Though we have now succeeded in aggregating suitability and confidence degrees using GCD operators, the interpretability of the results has not yet been discussed. It was silently assumed that the elementary uncertainty grades could be aggregated. This, of course, relies on the assumption that they all share the same semantics. As we have discussed in section 3.4, the interpretation of an elementary uncertainty grade actually depends on which defuzzification strategy is used. For an optimistic approach, a higher uncertainty grade is not a good sign, as it indicates that the actual suitability of the input could potentially be a lot lower than the estimated elementary suitability grade, whereas the opposite is true for a pessimistic approach. Clearly, aggregating an elementary uncertainty grade obtained by optimism with an elementary uncertainty grade obtained by pessimism is nonsensical, because the conflicting semantics regarding high values make it impossible to interpret whether or not a high value for the global uncertainty grade is actually good or bad.

The approach introduced in this sectino should really only be applied when

System	Attribute	Suit.	Negative	Positive
Car	Travel time	0.833	0.15	0.05
	Comfort	1.000	0.0	0.0
Train	Travel time	0.333	0.25	0.05
	Comfort	0.300	0.05	0.05
Airplane	Travel time	1.000	0.0	0.0
	Comfort	0.700	0.2	0.0

Table 4.13: Computed elementary suitability and bipolar uncertainty grades for the three systems.

System	Suitability	Negative	Positive
Car	0.914	0.077	0.026
Train	0.316	0.147	0.05
Airplane	0.840	0.107	0.0

Table 4.14: Computed global suitability and bipolar uncertainty scores for three sample systems.

all inputs are defuzzified using the same defuzzification strategy. However, this is undesirable, as it undermines one of the fundamental reasons why one would choose to defuzzify first, namely to have the freedom to express a different degree of tolerance towards uncertainty on different inputs. This is actually why we introduced the concept of bipolar uncertainty grades in section 3.4.3. By using the bipolar extension, we can choose any defuzzification strategy yet always end up with an elementary bipolar uncertainty grade for which the semantics are clear, and so we can aggregate them in a meaningful way.

Consider for example a set of defuzzified bipolar uncertainty values as shown in table 4.13. This table shows not only the defuzzified suitability of each attribute for each of the systems, but also the "negative" amount of uncertainty (area under the suitability distribution to the left of the estimated suitability, indicative of the possibility of an actually worse suitability than estimated) and a "positive" amount of uncertainty (area on the suitability distribution to the right of the estimated suitability, indicative of the possibility of an actually better suitability than estimated). The estimated suitability for the travel time of both the car and train systems and for the comfort of the airplane system is optimistic (there is more uncertainty attributed to the negative possible deviation than to the positive possible deviation of the bipolar uncertainty grades). Alternatively, the comfort of the train system was defuzzified with the center of mass approach.

The results of aggregating these bipolar uncertainty degrees, in a similar fashion as described above and separately combining negative degrees and positive degrees, are summarized in table 4.14. As these results show, the airplane system can really only underperform and disappoint, whereas the car system might surprise us in a positive (though small) way. As such, these results affirm our choice for the car system even more strongly.

Summary

We have shown how aggregation based on the GCD operator can be extended quite easily to not only aggregate suitability grades but also uncertainty grades into a single, global suitability-uncertainty duo.

All in all, we would nonetheless not recommend defuzzifying first if it can be avoided. However, as will become clear in the following section, direct aggregation is still rather unexplored and there are plenty of situations in which direct aggregation can not be carried out. In those cases, it might be interesting to defuzzify first, as it does provide a relatively simple solution for most practical cases. Yet, especially if the end-goal is *not* necessarily to create a global ranking in a decision support context, we suggest first trying to aggregate the distributions directly, in order to avoid losing information as much as possible.

4.4.2 Direct aggregation of suitability distributions

Now, let us cover how to aggregate suitability distributions directly. To that end, we propose to extend classical aggregation operators so they can be used to aggregate a collection of suitability distributions into a single, global suitability distribution, minimizing information loss. Note that this approach might be useful outside of a decision making context as well, because the result is not directly a ranking of the entities, but rather a global model for the (uncertain) suitability of an entity given the handling of (flexible) criteria. Though not directly rankable without defuzzification, a global suitability distribution still holds more information than any other aggregation result.

It is important to remember that a suitability distribution is essentially a model for possible worlds. It is not a characteristic function of a fuzzy set that describes the vague interpretation of a linguistic term. Whereas the latter associates each value from its domain with a degree of suitability, the former associates each value with a degree of possibility. As such, the operations for suitability distributions should not be confused with those defined on fuzzy sets given by Zadeh in [32]. This can easily be verified. For example, we want the complement of "certainly fully satisfied" to be "certainly not satisfied", not "fully uncertain". Hence, the operations for possibility distributions must be used [35]. These are devised to work on uncertain values and apply their operations on the domain of the fuzzy set, not on the membership grades of the elements.

Zadeh's Extension Principle

Operating in the interpretation of possibility distributions, it is possible to aggregate suitability distributions directly by using Zadeh's extension principle [35]. The extension principle asserts that any function that can be applied on elements of a universe of discourse (such as aggregators), can be extended in such a way that it can be applied on fuzzy sets defined on that universe. Using the following notation to express that $\widetilde{V}$ is a fuzzy set on elements u from the

universe of discourse $\mathcal{U}$

$$\widetilde{V} = \{(u_1, \mu(u_1)), (u_2, \mu(u_2)), \ldots, (u_n, \mu(u_n))\}$$

then a function $f : \mathcal{U} \to \mathcal{U}^*$, that transforms elements from one universe, $\mathcal{U}$, to new elements from another (or possibly the same) universe, $\mathcal{U}^*$, (e.g. the squaring operator $f : x \mapsto x^2$), can be extended such that

$$f(\widetilde{V}) = \{(f(u_1), \mu(u_1)), (f(u_2), \mu(u_2)), \ldots, (f(u_n), \mu(u_n))\}$$

Note that it might occur that some values u_i and u_j, $i \neq j$, collide under f (i.e. $f(u_i) = f(u_j)$. In that case, their corresponding membership grades μ_i and μ_j should be combined using a t-conorm (typically: the maximum t-conorm) into a single membership grade, for example: $\mu_{ij} = \min(\mu_i, \mu_j)$. In words, the grade of membership of a value y in $f(\widetilde{V})$ is computed as the maximal membership grade of all values u in $\widetilde{V}$ that collide under f. If $\mathcal{U}$ is a continuum, i.e.,

$$\widetilde{V} = \{(u, \mu(u)) \mid u \in \mathcal{U} : \mu(u) > 0\}$$

the extended form of f is written as

$$f(\widetilde{V}) = \{(f(u), \mu(u)) \mid u \in \mathcal{U} : \mu(u) > 0\}$$

In words, the extension principle states that applying a function on fuzzy sets results in a new fuzzy set whose elements can be derived from applying the function on the original elements, $u_1, u_2, \ldots$.

Example. Consider an uncertain number n that is given by the triangular distribution $(8, 10, 12)$ (around 10). In order to compute $2n$, we consider the doubling operator as a function of the form $f(x) = 2x$. Applying Zadeh's extension principle, we find that $2n$ is given by:

$$\begin{aligned} f(n) &= (f(8), f(10), f(12)) \\ &= (16, 20, 24) \end{aligned}$$

The outcome is described by a triangular distribution. This shows that doubling the uncertain value "around 10" leads to the uncertain value "around 20". Interestingly, the uncertainty in the description of "around 10" is also amplified by the doubling operator.

But what happens if f is n-ary (it takes multiple inputs, e.g. the addition operator $+$)? It is not yet explained how the outcome of $f(\widetilde{V}_1, \widetilde{V}_2)$ should be constructed, more precisely: how is the membership grade of element $f(u_1, u_2)$ related to $\widetilde{V}_1$ and $\widetilde{V}_2$? Zadeh proposed that the membership grade is determined by using a t-norm (i.e. the minimum t-norm) to combine the membership grades from the corresponding inputs:

$$\mu_{\widetilde{V}}(f(u_1, u_2)) = \min(\mu_{\widetilde{V}_1}(u_1), \mu_{\widetilde{V}_2}(u_2))$$

Example 2. Let $\widetilde{V}_1$ and $\widetilde{V}_2$ be fuzzy sets given by:

$$\widetilde{V}_1 = \{(2, 0.5), (3, 1), (4, 0.5)\}$$

(around 3) and

$$\widetilde{V}_2 = \{(3, 0.8), (4, 1), (5, 0.3)\}$$

(around 4) respectively. Applying Zadeh's extension principle to the addition operator, we find that $\widetilde{V}_1 + \widetilde{V}_2$ is equal to:

$$\begin{aligned}
\widetilde{V}_1 + \widetilde{V}_2 = \{&(2+3, \min(0.5, 0.8)), (2+4, \min(0.5, 1)), (2+5, \min(0.5, 0.3)),\\
&(3+3, \min(1, 0.8)), (3+4, \min(1, 1)), (3+5, \min(1, 0.3)),\\
&(4+3, \min(0.5, 0.8)), (4+4, \min(0.5, 1)), (4+5, \min(0.5, 0.5))\}\\
= \{&(5, 0.5), (6, 0.5), (7, 0.3),\\
&(6, 0.8), (7, 1), (8, 0.3),\\
&(7, 0.5), (8, 0.5), (9, 0.3)\}\\
= \{&(5, \max(0.5)), (6, \max(0.5, 0.8)), (7, \max(0.3, 1, 0.5)),\\
&(8, \max(0.3, 0.5)), (9, \max(0.3))\}\\
= \{&(5, 0.5), (6, 0.8), (7, 1), (8, 0.5), (9, 0.3)\}
\end{aligned}$$

which corresponds to "around 7".

Though the extension principle provides us with theoretical evidence that it is possible to aggregate suitability distributions, computing the actual analytical expression turns out to be hard in practice. Initial research on the topic of aggregating fuzzy sets in the context of decision making focusses on sampling elements from the domains of the fuzzy sets [1, 6], so as to be able to apply the aggregation on these sample points. Dong and Wong [5] were the first to implement an alternative solution that is based on α-cuts. An α-cut of a fuzzy set $\widetilde{V}$ is the strict set of all elements that have a membership grade of at least α in $\widetilde{V}$:

$$(\widetilde{V})_\alpha = \{v \mid \widetilde{V}(v) \geq \alpha\}$$

Similarly, a strict α-cut is given by:

$$(\widetilde{V})_{\hat{\alpha}} = \{v \mid \widetilde{V}(v) > \alpha\}$$

Strict α-cuts are typically only used to define the support of a fuzzy set (i.e. the strict 0-cut). Dong and Wong showed that their technique could provide a much better approximation of the true solution with only a few α-cuts, if they are chosen well. By using α-cuts, the computations are turned into interval analysis. The aggregation is carried out per α-cut and the resulting intervals are transformed back to a fuzzy set using the resolution identity [33, 34, 35]:

$$\widetilde{V} = \int_0^1 \alpha(\widetilde{V})_\alpha$$

The remaining question is which α-cuts should be considered? For trapezoidal functions, which is a fair limit to impose on ourselves given their practical importance, we would be tempted to rely on only the strict 0-cut and the 1-cut, which would correspond to performing element-wise aggregation. However, we will show in the following that this would lead to incorrect results. We will now give an overview of which results can be found by applying the technique of Dong and Wong on suitability distributions for common aggregation operators.

Basic operators

For the remainder of this section, let a_1 and a_2 denote *independent* (non-interactive) uncertain properties, c_{a_1} and c_{a_2} denote flexible criteria on these properties, and s_{a_1}, s_{a_2} are the respective suitability distributions, obtained by evaluating the criteria on the values for these properties of different entities.

Conjunction The conjunction of s_{a_1} and s_{a_2}, denoted by $s_{a_1 \wedge a_2}$, expresses the uncertainty over the degree to which a_1 and a_2 simultaneously satisfy respectively c_{a_1} and c_{a_2}. If we implement the conjunction with the Zadeh t-norm, then the least satisfied criterion is deciding in the aggregation. So, for each degree of satisfaction γ, $s_{a_1 \wedge a_2}(\gamma)$ can be expressed as follows: a_1 satisfies c_{a_1} to degree γ *and* a_2 satisfies c_{a_2} to a larger or equal degree, *or* a_2 satisfies c_{a_2} to degree γ *and* a_1 satisfies c_{a_1} to a larger or equal degree. The possibility that a_1 satisfies c_{a_1} to degree γ is given by $s_{a_1}(\gamma)$. The possibility that a_1 satisfies c_{a_1} to a degree that is *at least* γ can also be found from its suitability distribution as $\max_{\kappa > \gamma} s_{a_1}(\kappa)$. As such, it follows that:

$$\forall \gamma \in [0,1] : s_{a_1 \wedge a_2}(\gamma) = \max_{\kappa \geq \gamma}(\min(s_{a_1}(\gamma), s_{a_2}(\kappa)), \min(s_{a_1}(\kappa), s_{a_2}(\gamma)))$$

For trapezoidal suitability distributions, we can construct a more practical form. If s_{a_1} and s_{a_2} are trapezoidal suitability distributions, we can also write the conjunction in terms of α-cuts. For trapezoidal membership functions, it has been shown that each α-cut corresponds to exactly one interval [8]. As such, we can write:

$$(s_{a_i})_\alpha = [p_i, q_i]$$

Applying Zadeh's extension principle, we find that the conjunction takes the following form:

$$(s_{a_1 \wedge a_2})_\alpha = [\min(p_1, p_2), \min(q_1, q_2)]$$

Let the trapezoidal parameters specifying s_{a_i} be given by (g_i, h_i, k_i, l_i). Then we can rewrite the resolution identity as:

$$(s_{a_i})_\alpha = \begin{cases} [0,1] & \alpha = 0 \\ [h_i, k_i] & \alpha = 1 \\ [(1-\alpha)g_i + \alpha h_i, \alpha k_i + (1-\alpha) l_i] & elsewhere \end{cases}$$

Using this form for the conjunction, we obtain:

$$(s_{a_1 \wedge a_2})_\alpha = \begin{cases} [0,1] & \alpha = 0 \\ [\min(h_1, h_2), \min(k_1, k_2)] & \alpha = 1 \\ [f, n] & elsewhere \end{cases}$$

where

$$f = \min((1-\alpha)g_1 + \alpha h_1, (1-\alpha)g_2 + \alpha h_2)$$
$$n = \min(\alpha k_1 + (1-\alpha)l_1, \alpha k_2 + (1-\alpha)l_2)$$

For $\alpha \to 0$, we can derive that $f = \min(g_1, g_2)$ and $n = \min(l_1, l_2)$. It would be ideal if f could generally be written in terms of α, $\min(g_1, g_2)$ and $\min(h_1, h_2)$, because this would imply that the global suitability distribution is again trapezoidal. Unfortunately, the expression for f can not be reduced further, meaning that the aggregated suitability distribution is not necessarily trapezoidal and forcing us to investigate for possible breaking points.

We now know three things about the global suitability distribution:

1. Its support is given by $[\min(g_1, g_2), \min(l_1, l_2)]$.
2. Its core is given by $[\min(h_1, h_2), \min(k_1, k_2)]$.
3. It is not necessarily trapezoidal.

It is easy to find cases where the suitability distribution is not trapezoidal. For example, let $s_{a_1} = (0.1, 0.4, 0.5, 0.8)$ and $s_{a_2} = (0.2, 0.3, 0.6, 0.7)$. For $\alpha = 0.25$, f_1 (the f component of a_1) equals $7/40$ and f_2 (the f component of a_2 equals $9/40$, so $f_1 < f_2$. For $\alpha = 0.75$, $f_1 = 13/40$ and $f_2 = 11/40$, so $f_2 < f_1$. Due to the monotonic nature of the trapezoidal suitability distributions, there must be a certain α for which $f_1 = f_2$:

$$\begin{aligned} g_1 + \alpha(h_1 - g_1) &= g_2 + \alpha(h_2 - g_2) \\ \Leftrightarrow \alpha(h_1 - g_1 - h_2 + g_2) &= g_2 - g_1 \\ \Leftrightarrow \alpha &= \frac{g_2 - g_1}{g_2 - g_1 + h_1 - h_2} \end{aligned} \tag{4.17}$$

Unless $g_2 - g_1 = h_1 - h_2$ (i.e. when the slopes of the trapezoids are parallel), α exists. We are only interested in α-values from $]0, 1[$, because this would imply that the slopes of the two trapezoids actually intersect and that consequently the global suitability distribution has a breaking point in its slope. For our example, we find that $f_1 = f_2$ for $\alpha = 0.5$, so the global suitability distribution must show a breaking point for this value of α. An entirely identical reasoning applies to n. The entire example is visualized in Figure 4.13. This clearly shows that the global suitability distribution is *not* a trapezoid, and that the global suitability is furthermore not a point-wise minimum of the elementary

suitability distributions. Instead, the global suitability distribution is defined per α-cut and, per cut, traces the left-most elementary suitability distribution (i.e. the element-wise minimal suitability grade of the α-cuts of the elementary inputs).

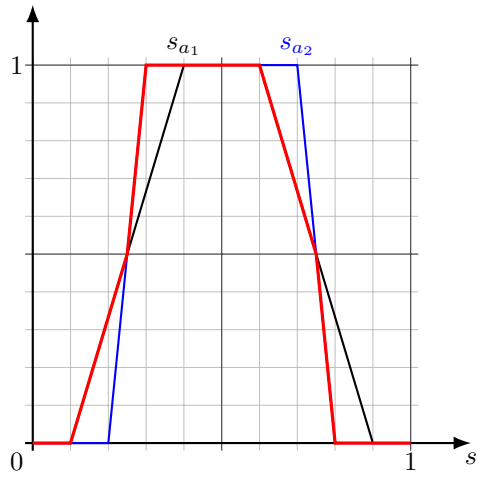


Figure 4.13: Conjunction of two trapezoidal suitability distributions, s_{a_1} and s_{a_2}, resulting in a non-trapezoidal global suitability distribution (in red).

Based on these observations, it is easy to verify that the conjunction of "certainly fully satisfied" and "certainly not satisfied" equals "certainly not satisfied", that the conjunction of "certainly fully satisfied" and "completely uncertain" is "completely uncertain", and lastly that the conjunction of "certainly not satisfied" and "completely uncertain" leads to "certainly not satisfied".

Disjunction The disjunction of s_{a_1} and s_{a_2}, denoted by $s_{a_1 \vee a_2}$, expresses the uncertainty over the degree to which either a_1 or a_2 satisfy respectively c_{a_1} or c_{a_2}. If we implement the disjunction with the max t-norm, then the most satisfied criterion is deciding in the aggregation. So, for each degree of satisfaction γ, $s_{a_1 \vee a_2}(\gamma)$ can be expressed as follows: a_1 satisfies c_{a_1} to degree γ *and* a_2 satisfies c_{a_2} to a smaller or equal degree, *or* a_2 satisfies c_{a_2} to degree γ *and* a_1 satisfies c_{a_1} to a smaller or equal degree. The possibility that a_1 satisfies c_{a_1} to degree γ is given by $s_{a_1}(\gamma)$. Alternatively, the possibility that a_1 satisfies c_{a_1} to degree *at least* γ can also be found from its suitability distribution as $\max_{\kappa > \gamma} s_{a_1}(\kappa)$. Hence, similar to the case of the conjunction, we find that:

$$\forall \gamma \in [0,1] : s_{a_1 \vee a_2}(\gamma) = \max_{\kappa \leq \gamma} (\min(s_{a_1}(\gamma), s_{a_2}(\kappa)), \min(s_{a_1}(\kappa), s_{a_2}(\gamma)))$$

Again, we can formulate a more practical form based on α-cuts if s_{a_1} and s_{a_2} are trapezoidal suitability distributions. By applying Zadeh's extension

principle, we find that the disjunction takes the form:

$$(s_{a_1 \vee a_2})_\alpha = [\max(p_1, p_2), \max(q_1, q_2)]$$

By the resolution identity, we have that the disjunction corresponds to:

$$(s_{a_1 \vee a_2})_\alpha = \begin{cases} [0,1] & \alpha = 0 \\ [\max(h_1, h_2), \max(k_1, k_2)] & \alpha = 1 \\ [f, n] & elsewhere \end{cases}$$

where

$$f = \max((1-\alpha)g_1 + \alpha h_1, (1-\alpha)g_2 + \alpha h_2)$$
$$n = \max(\alpha k_1 + (1-\alpha)l_1, \alpha k_2 + (1-\alpha)l_2)$$

For $\alpha \to 0$, we can derive that $f = \max(g_1, g_2)$ and $n = \max(l_1, l_2)$. Again, we find that f and n can not be reduced further, meaning that the disjunction of the trapezoidal suitability distributions s_{a_1} and s_{a_2} is not necessarily trapezoidal. Interestingly, but not surprisingly, the breaking point (if any) is the same as for the conjunction. For example, let again $s_{a_1} = (0.1, 0.4, 0.5, 0.8)$ and $s_{a_2} = (0.2, 0.3, 0.6, 0.7)$. Again we find that the breaking point accurs for $\alpha = 0.5$, but this time the support is given by $[\max(g_1, g_2), \max(l_1, l_2)]$. The full results of applying the disjunction on these distributions are visualized in Figure 4.14.

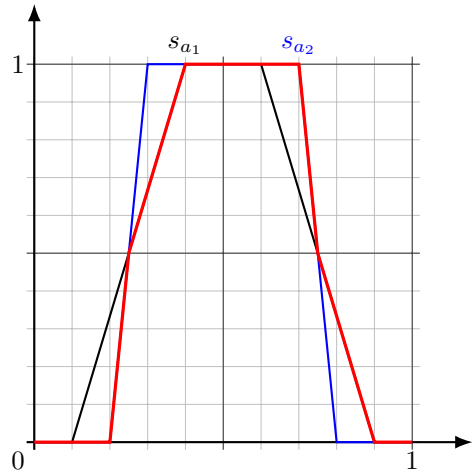


Figure 4.14: Disjunction of two trapezoidal suitability distributions, s_{a_1} and s_{a_2}, resulting in a non-trapezoidal global suitability distribution (in red).

Here, we have similar properties: the disjunction of "certainly fully satisfied" and "certainly not satisfied" is "certainly fully satisfied". The disjunction of "certainly fully satisfied" and "completely uncertain" is "fully satisfied". The disjunction of "certainly not satisfied" and "completely uncertain is "completely uncertain".

Complementation The complementation is perhaps a bit of an outsider in the discussion as it is not really an aggregator, yet we want to include the complementation as it corresponds to the logical inversion, which is not uncommon when constructing an aggregation structure. To define the complement of a suitability distribution, we once again revisit its interpretation. If $s_{a,\mathrm{c}}$ denotes the uncertainty regarding the degree to which a satisfies c, its complement $\bar{s}_{a,\mathrm{c}}$ should denote the uncertainty regarding the degree to which a does *not* satisfy c. Because c denotes suitability, we will use $1 - \mathrm{c}$ to denote *unsuitability*. If a value v_1 is fully suitable (i.e. $\mathrm{c}(v_1) = 1$), it should not be unsuitable. Letting $\bar{\mathrm{c}}$ be the complement to 1 [8, 7] of c, where $\bar{\mathrm{c}}(v)$ denotes the unsuitability of the value v, $\bar{\mathrm{c}}(v_1)$ should be equal to 0. In order to find $\bar{s}_{a,\mathrm{c}}$, one could evaluate a using $\bar{\mathrm{c}}$ (i.e. $\bar{s}_{a,\mathrm{c}} = s_{a,\bar{\mathrm{c}}}$) but this would require computing a second suitability distribution. This is not necessary if $s_{a,\mathrm{c}}$ has already been constructed, as $\bar{s}_{a,\mathrm{c}}$ can also be derived directly from $s_{a,\mathrm{c}}$, as follows:

$$\forall \gamma \in [0,1] : \ \bar{s}_{a,\mathrm{c}}(\gamma) = s_{a,\mathrm{c}}(1-\gamma) \tag{4.18}$$

To see why this is true, consider how $\bar{s}_{a,\mathrm{c}}$ would be constructed. First, the domain A would be partitioned by $\bar{\mathrm{c}}$ into classes of equal non-suitability, which is semantically opposite but essentially identical to the partitioning into classes of equal suitability (only in reversed order). Indeed, the degree of suitability of each partition class is instead mapped to a degree of non-suitability by taking its complement to 1. The possibility that a takes a value from any such class, stays the same. As such, the possibility that a dissatisfies c to degree κ is equal to the possibility that a satisfies $\bar{c}$ to degree κ, which is equal to the possibility that a satisfies c to degree $1 - \kappa$.

In terms of α-cuts, computing the complement of a trapezoidal suitability distribution s_a defined by (g, h, k, l) is given by:

$$\bar{s}_a = (1-l, 1-k, 1-h, 1-g)$$

Note that the order of the parameters is reversed in order to ensure that the parameters remain in increasing order. Geometrically, this complementation operator corresponds to a horizontal mirroring around the $s = 0.5$ line. An example of the complement operator is shown in Figure 4.15.

This implementation satisfies the following properties: the complement of "certainly fully satisfied" is "certainly not satisfied" (and vice versa) and the complement of "completely uncertain" is still "completely uncertain". This definition respects the definition of the complementation for possibility/necessity degrees.

Other operators

Technically, any operator can be extended so it can be used to aggregate suitability distributions by applying Zadeh's extension principle and using an α-cut approach. However, we have already shown that, even for basic operators, the global suitability distribution might show breaking points that were not present

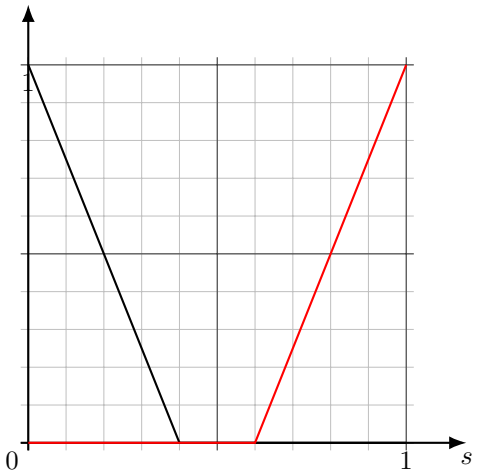


Figure 4.15: A suitability distribution and its complement (in red).

in the elementary suitability distributions. This has manifested itself in the fact that aggregating trapezoidal suitability distributions does not necessarily result in a trapezoidal global suitability distribution. Luckily, equation 4.17 provides a simple way to check for interesting α-cut values. If, given the parameters from two trapezoidal distributions (g_1, h_1, k_1, l_1) and (g_2, h_2, k_2, l_2), α falls in $]0, 1[$, then an interesting value for an α-cut is found. Note that this occurs when $g_1 < g_2 \wedge h_1 > h_2$ or $g_1 > g_2 \wedge h_1 < h_2$.

If the aggregation operator has the properties of a linear transformation, then the aggregation of only trapezoidal suitability distributions will result in a trapezoidal global suitability distribution. An example of this is the arithmetic mean.

Example. Let us take a look at what happens when we compute the arithmetic mean of two triangular, uncertain values. For once, we will not present them in their trapezoidal shortcut notation, because it is easier to transition into α-cut representation if they are shown in their regular, piece-wise definition. Let:

$$s_1(d) = \begin{cases} 4d & d \in [0, 1/4] \\ -4d + 2 & d \in [1/4, 1/2] \\ 0 & elsewhere \end{cases}$$

and

$$s_2(d) = \begin{cases} 4d - 2 & d \in [1/2, 3/4] \\ -4d + 4 & d \in [3/4, 1] \\ 0 & elsewhere \end{cases}$$

We can write these suitability distributions in terms of α-level sets:

$$(s_1)_\alpha = \begin{cases} [\alpha/4, (2-\alpha)/4] & \alpha > 0 \\ [0,1] & \alpha = 0 \end{cases}$$

and

$$(s_2)_\alpha = \begin{cases} [(2+\alpha)/4, (4-\alpha)/4] & \alpha > 0 \\ [0,1] & \alpha = 0 \end{cases}$$

Now we can compute the α-level sets of the global suitability distribution:

$$(s_g)_\alpha = \begin{cases} \left[\frac{\alpha/4+(2+\alpha)/4}{2}, \frac{(2-\alpha)/4+(4-\alpha)/4}{2}\right] & \alpha > 0 \\ [0,1] & \alpha = 0 \end{cases}$$

Besides the trivial case where $\alpha = 0$, we find that

$$\begin{aligned} (s_g)_\alpha &= \left[\frac{2+2\alpha}{8}, \frac{6-2\alpha}{8}\right] \\ &= \left[\frac{1+\alpha}{4}, \frac{3-\alpha}{4}\right] \end{aligned}$$

From this, we can derive the global suitability distribution:

$$s_g(d) = \begin{cases} 4d-1 & d \in [1/4, 1/2] \\ 3-4d & d \in [1/2, 3/4] \\ 0 & elsewhere \end{cases}$$

The proposed average global suitability distribution is uncertain but bound between 1/4 and 3/4. The only value that is fully possible is 1/2, which is the arithmetic mean of the fully possible points from the elementary suitability distributions. The worst (i.e. lowest) possible global suitability grade, 1/4, occurs when both elementary suitability distributions take their worst possible value and as such corresponds to the average of 0 and 1/2. The dual is true for the best (i.e. highest) possible global suitability grade. The example is visualised in Figure 4.16.

4.5 Conclusions

In this chapter, we have covered the topic of aggregation with broad sweeps. We have given an overview of existing aggregation techniques that are used in, among others, multi-criteria decision support systems to combine the results of elementary suitability grades into a single, global suitability grade, indicative of not only how well the preferences are satisfied but also the specified synergy between them. We have proposed and discussed two novel ways to aggregate suitability suitability distributions: by defuzzifying them first or by aggregating

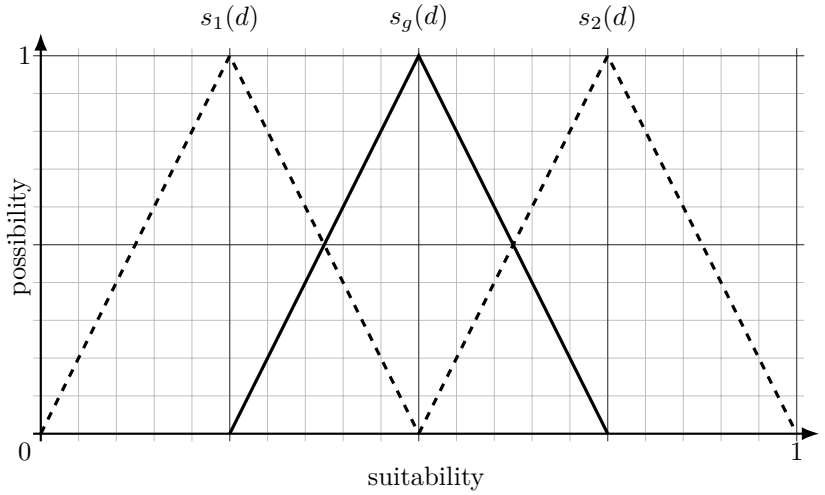


Figure 4.16: A visualisation of $s_1(d)$, $s_2(d)$ and their arithmetic average, $s_g(d)$.

them directly, through the extension principle. It is generally always best to aggregate the distributions directly, as less information is lost this way.

There is still a lot of work to be done in this area. We have only studied direct and suitability-uncertainty couple aggregation for binary operators, where there are only two inputs. Whereas defuzzifying first (without uncertainty indicators) provides a clear and easy path towards n-ary aggregation operators, the extension towards direct aggregation is less obvious, because one would have to check all combinations of suitability distributions for intersections. Even for trapezoidal suitability distributions, finding all possible intersections of the slopes is a problem of more-than-linear complexity.

Finally, we have so far always assumed that suitability distributions are possibilistic. However, as we have shown in section 3.5, suitability distributions can be applied in other settings, such as a probabilistic one. Though topics such as "defuzzification" and aggregation are better understood for those models (if they are among the common probability distributions, such as the normal distribution, the exponential distribution, etc.), it is not so straightforward how these different models of uncertainty can be harmonized into a single use-case. This is unfortunate, because it would definitely be plausible for real-world situations to combine both models (and perhaps even more models) to treat both stochasticity and the incompleteness of information.

Bibliography

[1] Sjoerd M Baas and Huibert Kwakernaak. "Rating and ranking of multiple-aspect alternatives using fuzzy sets". In: *Automatica* 13.1 (1977), pp. 47–58.

[2] Gustave Choquet. "Theory of capacities". In: *Annales de l'institut Fourier*. Vol. 5. Institut Fourier. 1954, pp. 131–295.

[3] Robin De Mol, Ana Tapia Rosero, and Guy De Tré. "An approach for uncertainty aggregation using generalised conjunction/disjunction aggregators". In: *16th World Congress of the International Fuzzy Systems Association (IFSA); 9th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT)*. Vol. 89. Atlantis Press. 2015, pp. 1499–1506.

[4] Guy De Tré, Jozo J. Dujmović, and Nico Van de Weghe. "Supporting spatial decision making by means of suitability maps". In: *Uncertainty approaches for spatial data modeling and processing : a decision support perspective* 281 (2010).

[5] W M Dong and F S Wong. "Fuzzy weighted averages and implementation of the extension principle". In: 21 (1987), pp. 183–199.

[6] Didier J Dubois. *Fuzzy sets and systems: theory and applications*. Vol. 144. Academic press, 1980.

[7] Didier Dubois and Henri Prade. "A review of fuzzy set aggregation connectives". In: *Information sciences* 36.1-2 (1985), pp. 85–121.

[8] Didier Dubois and Henri Prade. *Fundamentals of fuzzy sets*. Vol. 1. Springer Science & Business Media, 2000.

[9] Jozo Dujmović. "Aggregation Operators and Observable Properties of Human Reasoning". In: *Advances in Intelligent Systems and Computing*. Advances in Intelligent Systems and Computing 228 (2013). Ed. by Humberto Bustince et al., pp. 5–16. DOI: 10.1007/978-3-642-39165-1.

[10] Jozo Dujmović. "Andness and orness as a mean of overall importance". In: *IEEE International Conference on Fuzzy Systems* (2012), pp. 10–15. ISSN: 10987584. DOI: 10.1109/FUZZ-IEEE.2012.6250777.

[11] Jozo Dujmović. *Soft computing evaluation logic: the LSP decision method and its applications*. Wiley-IEEE Computer Society Press, 2018.

[12] Jozo J Dujmovic. "Continuous preference logic for system evaluation". In: *IEEE Transactions on Fuzzy Systems* 15.6 (2007), pp. 1082–1099.

[13] Jozo J Dujmović. "Extended continuous logic and the theory of complex criteria". In: *Publikacije Elektrotehničkog fakulteta. Serija Matematika i fizika* (1975), pp. 197–216.

[14] Jozo J Dujmović. "Partial absorption function". In: *Publikacije Elektrotehničkog fakulteta. Serija Matematika i fizika* (1979), pp. 156–163.

[15] Jozo J Dujmović. "Weighted conjunctive and disjunctive means and their application in system evaluation". In: *Publikacije Elektrotehničkog fakulteta. Serija Matematika i fizika* (1974), pp. 147–158.

[16] Jozo J Dujmovic, Guy De Tré, and Suzana Dragicevic. "Comparison of Multicriteria Methods for Land-use Suitability Assessment." In: *IFSA/EUSFLAT Conf.* Citeseer. 2009, pp. 1404–1409.

[17] Jozo J Dujmović and Henrik Legind Larsen. "Generalized conjunction/disjunction". In: *International Journal of Approximate Reasoning* 46.3 (2007), pp. 423–446.

[18] Jozo Dujmović and Guy De Tré. "Multicriteria methods and logic aggregation in suitability maps". In: *International Journal of Intelligent Systems* 26.10 (2011), pp. 971–1001.

[19] János Fodor. "Left-continuous t-norms in fuzzy logic: An overview". In: *Acta Polytechnica Hungarica* 1.2 (2004).

[20] Michel Grabisch. "Fuzzy integral in multicriteria decision making". In: *Fuzzy sets and Systems* 69.3 (1995), pp. 279–298.

[21] Michel Grabisch. "K-order additive discrete fuzzy measures and their representation". In: *Fuzzy sets and systems* 92.2 (1997), pp. 167–189.

[22] Michel Grabisch and Christophe Labreuche. "A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid". In: *Annals of Operations Research* 175.1 (2010), pp. 247–286.

[23] Erich Peter Klement, Radko Mesiar, and Endre Pap. *Triangular norms*. Vol. 8. Springer Science & Business Media, 2013.

[24] Christophe Labreuche and Michel Grabisch. "The Choquet integral for the aggregation of interval scales in multicriteria decision making". In: *Fuzzy Sets and Systems* 137.1 (2003), pp. 11–26.

[25] Toshiaki Murofushi, Michio Sugeno, and Katsushige Fujimoto. "Separated hierarchical decomposition of the Choquet integral". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 5.05 (1997), pp. 563–585.

[26] Michio Sugeno. "Theory of fuzzy integrals and its applications". In: *Theory of Fuzzy Integrals and Its Applications* (1975).

[27] Michio Sugeno, Katsushige Fujimoto, and Toshiaki Murofushi. "A hierarchical decomposition of Choquet integral model". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 3.01 (1995), pp. 1–15.

[28] Ana Tapia Rosero and Guy De Tré. "Evaluating relevant opinions within a large group". eng. In: *6th International Conference on Fuzzy Computation Theory and Applications, Proceedings*. Ed. by António Dourado, José Cadenas, and Joaquim Filipe. Rome, Italy: SciTePress, 2014, pp. 76–86. ISBN: 9789897580536.

[29] Zhenyuan Wang and George J Klir. *Fuzzy measure theory*. Springer Science & Business Media, 2013.

[30] Ronald R Yager. "On ordered weighted averaging aggregation operators in multicriteria decisionmaking". In: *IEEE Transactions on systems, Man, and Cybernetics* 18.1 (1988), pp. 183–190.

[31] Ronald R. Yager. "On a general class of fuzzy connectives". In: *Fuzzy Sets and Systems* 4 (1980), pp. 235–242. ISSN: 0300-0508. DOI: 10.3138/ptc.2009-09-s1.

[32] L A Zadeh. "Outline of a new approach to the analysis of complex systems and decision processes". In: *IEEE Transactions, SMC- 3* 1 (1973), pp. 28–44. ISSN: 0018-9472. DOI: 10.1109/TSMC.1973.5408575.

[33] Lot A Zadeh. "Fuzzy languages and their relation to human and machine intelligence". In: *Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A Zadeh*. World Scientific, 1996, pp. 148–179.

[34] Lotfi A Zadeh. "Similarity relations and fuzzy orderings". In: *Information sciences* 3.2 (1971), pp. 177–200.

[35] Lotfi A Zadeh. "The concept of a linguistic variable and its application to approximate reasoning". In: *Information sciences* 8.3 (1975), pp. 199–249.

Chapter 5

Indexing (Uncertain) Data

From a pragmatic point of view, a database is a digitized persistent storage of information and a query corresponds to a representation of a model solution to a certain problem. Though the database might not contain an entity that is identical to the model solution, evaluating the query comes down to asking a question to the dataset in order to retrieve those entities that best resemble the described model solution.

On a technical level, the data describing the entities are represented by bitstrings that are scattered across the different sectors of a hard disk and a query is an algebraic expression that is evaluated for each entity in the database. In order to be able to evaluate the query expression for a certain entity, its data must be accessible to the computer's central processing unit (CPU). Therefore, hard disk sectors containing database information must be copied ("loaded") into main memory, which is a costly (slow) operation. Database sectors that are loaded into main memory are referred to as *database buffers*. The amount of database buffers that is used depends on how much main memory is available. Typically, there will always be at least two buffers in use, so that the CPU can keep processing data from one buffer while the other is being loaded or flushed (written back to the hard disk after being modified). Figure 5.1 gives a schematic overview of the entire memory layout.

In order to be able to ensure that all entities that satisfy the criteria of a query are found, a database system without extra facilities must load and process *all* disk blocks containing entities. For some of these disk blocks, evaluating the query expression will lead to the conclusion that no entities in that block satisfy the query criteria. It would be better to avoid loading such disk blocks, because loading them (and evaluating the entities that they contain) is nothing but a waste of time. The smaller the fraction of the entities in the database that are actually a good match for the query criteria, the more time is typically wasted by exhaustively checking all sectors containing entities.

When evaluating a query, a database may use one or more indices in order to avoid loading disk blocks that do not contain entities relevant to that query. An index is essentially a data structure that serves as a way to look up entities

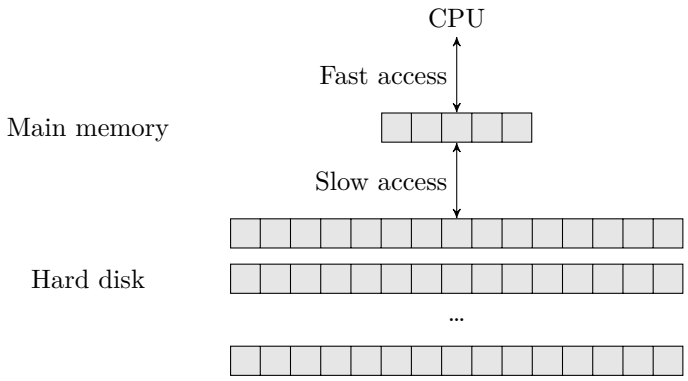


Figure 5.1: A visual illustration of the memory layout of a computer. The database is stored across different sectors of the hard disk and the buffered parts are kept in main memory.

by one or more of its attribute values. When uncertainty is allowed in the database, traditional indexing techniques can no longer be applied due to their strong reliance on precise attribute values. Instead, new indices, specifically designed to deal with fuzzy data, are required.

In this chapter, we propose a new index for this specific purpose called Interval B^+-tree (IBPT). The contents of this chapter are currently being reviewed after being submitted for publication in Fuzzy Sets and Systems.

The structure is as follows: in section 5.1, the basic concepts of indexing are explained. It also serves as a reference regarding common important indexing techniques and provides some background regarding B^+-trees and its current adaptations. Section 5.2 describes the preselection principle, a common methodology for creating indices for fuzzy data. In section 5.3, we introduce the IBPT indexing technique. Section 5.4 describes a set of experiments that were conducted. Finally, section 5.5 summarises our findings.

5.1 Indexing

A key concept in speeding up select query evaluation is *indexing*. A database system may maintain one or more indices for the data it manages. A conventional index is essentially nothing more than a minimized, highly structured and typically ordered copy of a specific set of attribute values of a set of entities of the same entity type, accompanied by pointers to the addresses of the hard disk blocks where the corresponding full entity data can be found. Database systems use indices to retrieve entity data by attribute values.

Indices are usually built with the values of a single attribute, but in some cases it is warranted to create a so-called *composite* index that combines values from several attributes in a single data structure. Composite indices are generally less flexible but can offer a performance boost over maintaining sev-

eral separate indices when it comes to evaluating queries that frequently impose criteria on the same collection of attributes.

Adding indices requires additional storage space on top of that which is required to store the data itself. In adition, indices incur overhead when evaluating a query (loading the index from the hard disk is (almost) always an unnecessary load operation because the index does not contain actual entity data besides the indexed attribute values that are copied). Nevertheless, these downsides are well outweighed by the positive impact on select query evaluation.

Indices are (usually much) smaller than the original data set, not only because they only contain a fraction of the original data but also because the database system can exploit spatial locality when manipulating the index. The latter implies that database systems can guarantee that entire disk blocks are efficiently packed with index data. We will illustrate this with an example.

Example. Consider a collection of 5000 entities, each consisting of 12 attribute values and spread across different addressable hard disk blocks that can each store 1024 bytes of data. Under the assumption that each entity takes up exactly 48 bytes (4 per attribute), one block can theoretically contain up to 21 entities. This would imply that one would need a bare minimum of 239 blocks to store all entities, but assume that, due to fragmentation, the entities are instead scattered across 750 different bocks. Lastly, assume that at least one of the 12 attributes is unique (i.e. each entity has a different value for this attribute). Given a specific value for this attribute, the goal is to obtain the data for the remaining attributes of the corresponding entity. Without index, this would require reading 375 bocks on average, 1 at best and 750 at worst. An index on this attribute would consist of copies of the 5000 unique values and 5000 block address locations, grouped in value-block address couples. Given that a block address is 5 bytes long itself (and the value is 4 bytes large), a block can contain up to 113 value-block address couples of each 9 bytes large. Because the database knows all values and addresses at the time it builds the index, it can guarantee that, when saving the index to disk blocks, each block will in fact contain 113 of these value-block address couples. Doing so, the index would require exactly 45 blocks to store. Retrieving the data of a single entity given its unique value now only requires reading 24 blocks on average (23 from scanning the index and 1 for loading the actual block containing the data), 2 at best and 46 at worst. In this example, the index takes up about 6% as much space as the data set but speeds up query evaluation by a factor of 16.

Before going into the details of specific indices, it should be noted that we assume that the values that need to be indexed contain no duplicates (i.e. no two separate entities have the same value for the same attribute). Even if there were duplicates, the structure of the indices we are about to discuss would be identical. The only difference is that an additional indirection layer would have to be added. This indirection layer works as follows: for each unique attribute value, a disk block is constructed containing a list of pointers to all blocks containing entities with this attribute value. Typically, a separate disk block

is used per attribute value. Should there be so many entities that all their combined addresses exceed the size of a single disk block, a chaining mechanism can be applied to link multiple blocks in sequence. Figure 5.2 visualizes the indirection layer.

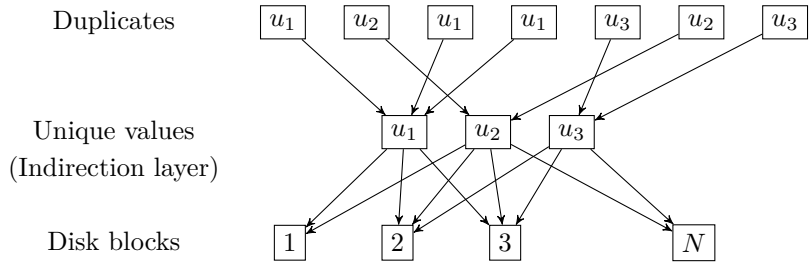


Figure 5.2: Visualization of the effects of adding an indirection layer, necessary when creating an index on datasets that contain duplicate values.

Any index is then built using only the unique attribute values, and instead of pointers to associate the values to disk block addresses containing the entity data, a pointer to block of the indirection layer containing the addresses of all entities with that value is stored.

Creating an index on a non-unique attribute will thus unavoidably add (at least) one extra disk block transfer. However, in comparison to an index on a unique attribute for the same dataset, an index on a non-unique attribute will typically be smaller due to the lower amount of distinct values, resulting in fewer leaf nodes. If this would result in the tree having (at least) one fewer level, then the cost of having to load the additional disk block from the indirection is already mitigated.

5.1.1 Composite indices

Most practical indices are built using only values from a single attribute. This promotes flexibility due to composability: if a query poses constraints on multiple attributes, the corresponding indices (if they exist) can be evaluated separately and the final result set can be computed by combining the intermediary results. For an entity type with n attributes, this would imply creating (at most) n separate indices in order to be able to efficiently cope with any ad hoc query. For conjunctive queries, combining the intermediary results comes down to some form of intersection. However, intersection implies that the final result can never be larger than any of the intermediary result sets. As such, constructing the intermediary result sets only to eliminate parts of them later implies that a portion of disk blocks are loaded unnecessarily. Recall that the entire point of indexing is to avoid loading disk blocks that do not contain relevant entities. In order to optimize the construction of the result set for queries that conjunctive combine multiple criteria, one might construct a composite index that uses values from multiple attributes.

For an entity type with n attributes, one could construct at least 2^n composite indices (more if different orders are considered per attribute). This exponential relation indicates that exhaustively creating composite indices in order to support ad hoc queries is not feasible. Instead, it is recommended to only create composite indices for specific sets of attributes that are expected or known to be used together frequently in query criteria.

5.1.2 Multi-level indexing

Indices based on sortable attributes can be further optimized by adding additional *levels* (or layers) to the index. The second layer of an index is actually just another index which treats the first index as data. Sortability is important here, because it is due to the fact that the data is sorted and efficiently packed in disk blocks by a first index, that high levels of compression can be achieved. The trick is namely to not include every indexed attribute value in the second layer, but only the first value per block. The values that are included are sometimes called *sentinel* values. When a non-sentinel value is sought after in the second level of the index, the sentinel values provide sufficient information to deduce in what first level disk block the non-sentinel value can be found.

To continue the previous example, the second layer of the index would also consist of value-block address couples but because the 5000 original values are sorted across 45 first-level blocks by the index, only 45 sentinel values need to be included in the second layer of the index. As such, the second layer must store 45 entries of 9 bytes large, which luxuriously fits within a single block. The final index is visualized in Figure 5.3

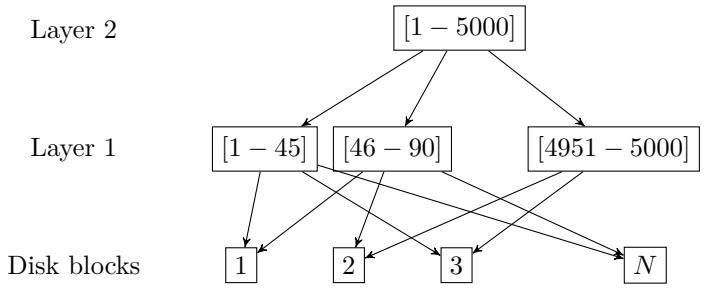


Figure 5.3: Visualization of the logical structure of a layered index, showing a two-layer index of 5000 values copied from entities that are fragmented across N hard disk blocks.

Note that this layering process can be (and usually is) repeated until the highest layer of the index fits within a single disk block. In order to retrieve the data of a particular entity based on a given attribute value that uniquely identifies the entity, the database system would start by loading the block containing the highest layer of the index. Based on the sentinel values in this layer, the database system can derive precisely which block from the next layer contains

the given value. That block is subsequently loaded into memory and the process repeats iteratively until, finally, the block containing the actual value is found. At that point, the disk block that is referenced by the address coupled to the given value is loaded and the entity data is retrieved. In our example, fetching the data of the entity would now take three load operations: one for loading the root block of the index, containing the entire second layer, one for loading the block of the first layer of the index that contains the value, and a final load operation of the block containing the data.

5.1.3 Disadvantages of indexing

Indices can speed up query evaluation, but have drawbacks, too.

1. When evaluating a select query, a database can really only rely on indices that are built on the attributes that are actually used in the filter predicate of the query. Queryies that specify a filter on a non-indexed attribute do not benefit from indices. In comparison, we can use a phone book to find phone numbers by region and last name, but not by first name. Listing the phone numbers for all people called Steve would require checking each entry in the phone book.

2. Each index structure must be maintained in order to stay usable and useful. Each time new data is inserted or existing data is updated or deleted, all relevant indices must be updated. This overhead has a negative impact on the rate at which database systems are able to process elementary operations that modify data and scales with the amount of indices that are created.

3. Most indices rely on some sort of order of the indexed attribute values and as such are only applicable for select queries with a simple filter predicate. In situations where the filter predicate expresses a conjunction of multiple domain filters on different attributes, a database system will usually first evaluate the different filters independently and then compute the intersection of the individual results, returning only those entities that occur in each independent result set. Though some index structures can be created on multiple attributes (typically the case for spatial data), this is generally not the case.

4. Indices are generally geared towards speeding up queries whose result set is small. It is usually assumed that small result sets are the result of filters that represent equality tests. Consequently, equality tests frequently form a fundamental part of the index lookup strategy. Though it is rather uncommon, it can happen that a small result set does not necessarily correspond to a particular (combination of) attribute value(s). However, indices typically do not help in these cases. An example of simple select queries that do not benefit from indices are those that use an inequality filter, especially if the attribute values are (very) unevenly distributed.

Though it might seem so at first glance, creating a separate index for each attribute (and each combination thereof) is no solution in most practical cases, because the overhead per elementary operation scales with the amount of indices. Besides, the amount of possible indices grows exponentially with the amount of attributes per entity type, as opposed to the data set itself, which grows only linearly in the amount of attributes, meaning that one rapidly could require more space to store indices than the actual data itself. In conclusion, indices should be created thoughtfully with the intention to support queries that are expected to be submitted frequently.

5.1.4 Indices using balanced trees

The first family of indexing techniques that we will introduce is based on *graphs*. A graph is a mathematical data structure that consists of nodes (or vertices) and edges. Each edge connects exactly two nodes and can be unidirectional (directed) or bidirectional (undirected). The nodes that are directly connected to a certain node (by means of an edge) are called its neighbours. A path is defined as a sequence of joining (i.e. sharing a common node) edges. The length of a path is given by the amount of edges it contains. Typically, there can be multiple paths connecting any pair of nodes A and B (or there could be none, which is more typical for directed graphs). An example of a graph is given in Figure 5.4.

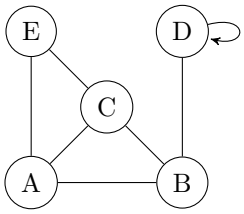


Figure 5.4: An example of a graph consisting of five vertices (A, B, C, D and E) and a few edges (including a loop at D).

Trees form a special subset of graphs in which any two nodes are connected by one and only one path. When used in the context of indexing, the edges are directed, and no loops are allowed (edges that connect a node to itself). With the exception of the root node, each node must have exactly one incoming edge (coming from its so-called *parent* node) and zero or more outgoing edges (going to its so-called *child* nodes). The root is special in the sense that it is the only node without parent. Nodes that have no outgoing edges are called leaf nodes. An example of a tree is given in Figure 5.5.

Balanced trees are trees that are structured in such a way that all paths from the root of the tree to the leaf nodes have the same length. Based on what has already been said about multi-level indices in section 5.1.2, a balanced tree is an ideal data structure to represent a multi-level index: its nodes are

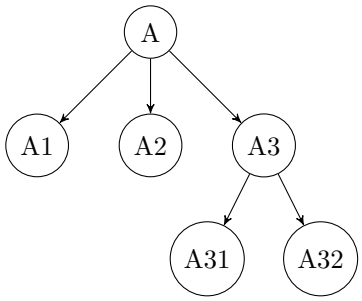


Figure 5.5: An example of a directed tree consisting of three levels (counting the root, A). The leaf nodes are A1, A2, A31 and A32.

mapped to the logical representation of hard disk sectors and the highest level of the index is stored in the root of the tree. In a tree-based index, the root is typically referred to as the 0th level. All nodes that can be reached in one hop from the root form the 1st level, and so on. An example of a balanced tree is given in Figure 5.6.

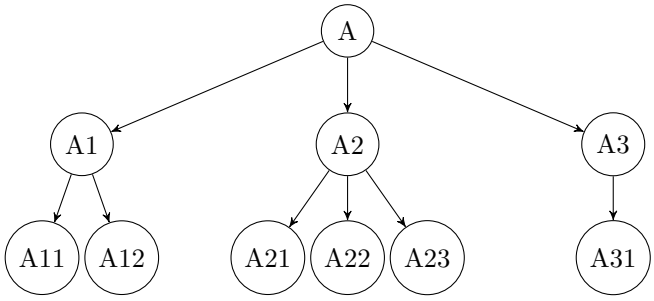


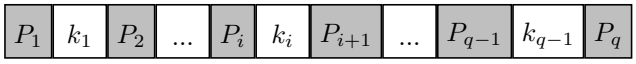
Figure 5.6: An example of a balanced tree.

Keeping the tree balanced ensures that, on average, only a minimal amount of disk blocks must be read in order to retrieve an arbitrary value. Though it would be possible to construct a more efficient index in case there is prior knowledge regarding which values are queried more frequently than others (e.g. by ensuring these values can be reached with fewer disk block reads than other values), balanced trees are especially useful in the (often realistic) case when there is no such prior knowledge. Balanced trees are also exceptionally resilient to data being added or removed.

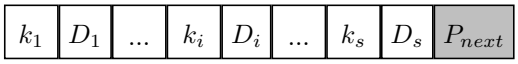
B^+-trees

A B^+-tree (BPT) is a special balanced tree which stores all its values in its leaf nodes. Some values are duplicated to internal nodes to serve the purpose of

being able to navigate through the tree efficiently. Of all indexing techniques for numerical values, the BPT is among the most efficient and widely used on block-based data stores (such as hard disks) because they are balanced and their structure matches block-based persistent data stores very well (i.e. each node in the tree corresponds to a disk block) [1]. In addition, they have generally good support for range queries by storing the values in an ordered fashion and maintaining a linked list that traverses them in this order. The key to their efficiency lies in how data are structured within nodes. Figure 5.7 illustrates the general structure of BPT nodes.



BPT internal node with q child nodes.



BPT leaf node containing s values.

Figure 5.7: Graphical examples of a BPT internal node (top) and a BPT leaf node (bottom).

A BPT is defined by its order (or branching factor), b, which indicates the maximal fan out (amount of children) that internal nodes can have. The order of a BPT can theoretically be chosen but is practically determined by the size of the disk blocks in such a way that each node of the index corresponds to a disk block. The precise way to calculate b is given later.

An internal node of a BPT contains $q-1$ ordered key values $k_i, i = 1, ..., q-1$ and q pointers $P_i, i = 1, ..., q$ to q subtrees $S_i, i = 1, ..., q$ such that every search key value k in S_i satisfies $k_{i-1} \leq k < k_i$. Consequently, S_1 contains the smallest value in the entire index, whereas S_q contains the largest value. Keeping q between $\lceil b/2 \rceil$ and b (for all nodes except the root node) offers an efficient trade-off between optimally using the available space in each disk block and yet also reserving sufficient space in order to be able to perform insert and delete operations without triggering many rebalancing operations. Many database systems force this behavior and call it the *fill factor constraint*.

Each BPT leaf node contains s key values $k_i, i = 1, ..., s, k_1 < k_2 < ... < k_s$, s data pointers $D_i, i = 1, ..., s$ (which point to the disk blocks containing the entities corresponding to these keys) and a single tree pointer P_{next}. The extra tree pointer P_{next} allows rapid sequential traversal of all leaf nodes in ascending key order. Though this technically goes against the theoretical definition of a tree (in the sense that, due to this linked list, there are now multiple possible paths connecting the root and the leaf nodes), this extra tree pointer drastically improves the evaluation speed of range queries (i.e., those queries that request all entities with a value from a specified range). Because of the P_{next} pointers, one only needs to look up the lowest value from the range and then use the

linked list to retrieve leaf node after leaf node directly until the highest value from the range is found (or surpassed). This way, the tree is only traversed once from root to leaf, and only the values from the specified range that are actually present in the leaf nodes are tested. For one leaf node (the one that terminates the linked list), P_{next} is a null-pointer. In a BPT, s is kept between $\lceil r/2 \rceil$ and r (the maximum amount of keys that fit in a leaf node). The only exception to this is when the tree consists of only one node (in which case the root node is a leaf node). Other variants of balanced trees might use a different optimum, e.g. B*-trees try to keep s around $2/3$.

Note that r is not necessarily equal to b. If $||B||$ is the size (in bytes) of a disk block, $||k||$ is the size (in bytes) of a key value and $||P||$ is the size (in bytes) of a tree pointer, b can be found as the largest integer value for which the following equation holds:

$$||B|| \geq b.||P|| + (b-1).||k|| \tag{5.1}$$

Further assuming $||D||$ is the size (in bytes) of a data pointer, r is the largest integer value for which the following equation holds:

$$||B|| \geq r.(||D|| + ||k||) + ||P|| \tag{5.2}$$

In practice, tree and data pointers typically take up the same amount of space ($||D|| = ||P||$), as they both represent a disk block address. Under these assumptions, it follows that $r = b - 1$ and that internal and leaf nodes are structurally indistinguishable. Going forward, we assume this is the case.

An example BPT of order 5 is shown in Figure 5.8.

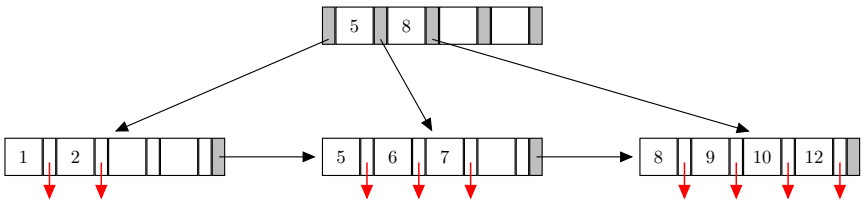


Figure 5.8: Example BPT of order 5 after inserting (in order): 8, 5, 1, 7, 2, 12, 10, 6 and 9.

We will briefly describe the base algorithms that are supported by a B$^+$-tree. For more details concerning these operations, we refer to [1].

Look-up. Searching for a value in a BPT is an iterative process of visiting nodes. It starts at the root and ends when a leaf node is reached. When visiting an internal (i.e. non-leaf) node, the subtree that could potentially contain the look-up value is found and the corresponding child node is visited next. When finally a leaf node is visited, two cases can occur: either the leaf node contains

the look-up value, or it does not. In case of the latter, the algorithm ends with no result. In the former case, the disk block pointer associated to the look-up value indicates the disk block where (the start of) the full entity is stored. That disk block is loaded and returned, also terminating the algorithm. Retrieving an entity indexed by a BPT of t levels requires the loading of $t + 1$ disk blocks: one for every level of the tree (root included) and one for the final disk block containing the entity data.

Insertion. In order to insert a new value into the index, the BPT is first traversed in order to identify the leaf where the value should be placed. If the leaf is not full, the value is inserted and the insertion algorithm stops. If the leaf is full, the node "overflows" and is split in two new leaf nodes and the keys are redistributed across them. Typically the first $\lceil r/2 \rceil$ values are stored in one leaf and the remaining values are stored in the other leaf. The lowest value of the other leaf is copied to the parent, which triggers an insert into an internal node. Inserting into internal nodes is similar to inserting in leaf nodes but with one subtle difference: when overflowing, the lowest value of the other node is *moved* to the parent instead of copied. This ensures that each key value can only occur once among the internal nodes. In case the root splits, a new parent node is created and the tree grows by one level. This new parent becomes the new root of the tree.

Inserting a value has a limited complexity. At worst, an overflow occurs at every level, meaning there are as many splits as there are levels in the tree, which is logarithmic in the amount of keys that are indexed by the tree.

Deletion. In order to delete a value, the leaf that would contain the value is first identified by traversing the BPT, similar to look-up and insertion. If the leaf contains the value, it is deleted. In doing so, it could occur that the leaf contains fewer keys than $\lceil b/2 \rceil$ ("underflow"). In order to restore the fill factor constraint, the keys of neighbouring leaves are redistributed. First, the immediate neighbours are consulted to see if any of them can donate a value without themselves underflowing. If neither neighbour has sufficient values, the leaf is merged with either of its neighbours into a single leaf, absorbing a value from their parent in the process. This process is repeated for the parent node until each node satisfies the minimal fill factor constraint.

The complexity of deleting a value is similar to that of inserting a value and is hence logarithmic in the amount of entities that are indexed by the BPT.

Update. Updating value X to Y is done by first deleting X and then inserting Y. The complexity of updating a value is also logarithmic in the amount of keys that are indexed by the BPT.

Composite BPT

Up to here, we have only covered trees built on a single attribute. However, BPTs can also be used to build composite indices. For the resulting so-called

composite BPT (CBPT), the order in which the attributes are specified is important. The essential idea behind CBPT is to sort the indexed entities by a certain attribute first, and to use the next attribute as a tiebreaker, and so on. For an index on n attributes, one can create (at least) $n!$ different composite indices (even more if the indexed attributes can be sorted according to several orders). For example, one could sort people's names by ascending last name first, and ascending first name second, or the other way around.

Building a CBPT index is similar to building a regular BPT, but the keys are composed of the concatenation of all of the indexed attribute values. Consequently, CBPTs have a typically lower branching factor. Everything else (procedures for look-up, insert and delete) remain the same.

When evaluating a query, a database is more likely to use a composite index if the query poses constraints on only the first or at least all attributes that are used in the index. This makes a composite index generally less applicable than a collection of separate BPT indices Composite indices can be used when only the first attribute is subject to filtering due to the fact that a CBPT uses this attribute to build its internal nodes. Nonetheless, a composite index would be strictly less efficient than a regular BPT because it is necessarily larger by having to store information on the other attributes, too. In cases when all relevant attributes are included in a query, it is easy to verify that a composite index is more efficient than having a separate index for each attribute. A collection of separate BPT indices would not only take up more storage space, but finding the result set would require evaluating each of them individually and combining each partial result sets. For a conjunctive query, this would result in having to compute the intersection, a process during which many false positives (matching one filter but not all of them) would be loaded into memory, which is a waste. In case a CBPT is used, only one index structure is traversed, resulting in far more efficient disk block reads. All records in the leaf node are tested to see if they match the query constraints. Then, the P_{next} list is used to traverse the leaf nodes until the first record not matching the constraint on the first attribute is found. Due to the structure of the index, all following records can no longer satisfy the first constraint and thus it is guaranteed that all potentially matching records have been tested. This procedure is guaranteed to load a minimal amount of disk blocks containing no relevant entities.

For a certain order, it should be noted that a CBPT will be "larger" than a regular BPT. Larger here means that there are either more nodes or that the nodes are more filled. In other words: a CBPT of order b, built on 50000 entities will take up more space than a BPT of order b, built on 50000 entities. This is because the leaf nodes must store more bytes per indexed record for each extra attribute that is added to the index. Nevertheless, this will only have an impact on query efficiency in case this would result in an extra level being added to the tree, meaning that each query would then require an additional disk block to be read. In any case, a CBPT will take up less space than a collection of separate BPTs for each of the attributes, and particularly queries that make use of all indexed attributes will executed significantly more efficiently with a CBPT index.

5.1.5 Alternative techniques

Bitmap indexing

Bitmap indices [17] are a fairly new (compared to tree-based indexing techniques) way of indexing data that is based on bitmaps. A bitmap is essentially a bit matrix for a single attribute with a finite domain where each value from the domain translates to a column in the matrix, and each entity in the dataset corresponds to a row. As such, each entity corresponds to a bit string which can be seen as a mask that identifies its value. Bitmaps are interesting because of multiple reasons:

- They significantly speed up query evaluation.
- They do not depend on the existence of a total order on the domain of the indexed attribute.
- Due to their bitwise nature, they are very efficient at evaluating logical combinations of values, such as "and", "or", "not", "xor",....

Bitmaps perform very well for ad hoc querying including multiple attributes, even without composite indices. The reason this is true is because, unlike with other indexing techniques, the results of individual bitmap index lookups can be combined before any disk blocks are loaded. Indeed, the bitmasks for the potential entities can be fetched and combined logically before any blocks have to be loaded, the entire combination logic can be performed at the level of the bitmasks itself.

Bitmaps are however less efficient when it comes to maintenance operations such as inserting, updating or deleting data. Whereas adding new entities is easy, adding new columns is not. Especially in case that the length of the bitmask would be extended beyond a multiple of the length of the machine *word* (the unit of operation for processors, typically 32 or 64 bit), updating the bitmap is a costly operation. Hence, bitmaps are mostly found in read-only systems that focus on rapid querying, like data warehouses, online analytical processing (OLAP) and online transaction processing (OLTP) applications. Furthermore, the size of bitmaps scales with the amount of distinct values in the attribute's domain, and as such are typically used for attributes with few distinct values (e.g. gender). Though it might seem to be implied that bitmaps are most efficient for attributes with a small amount of distinct values, it has been shown that bitmaps can also be highly performant for attributes with a large domain (e.g. unique value per row), despite their quickly increasing size[1].

Spatially, each bitmask denoting an entity is preferrably kept in a single disk block. The amount of such entity bitmasks that can be fit within a single disk block depends on the length of the bitmask, which is determined by how many attributes and attribute values are indexed. The *fragmentation* of a bitmap (i.e. the fraction of "wasted" space in each disk block) depends on how close the length of each bitmask is to a multiple of a machine word. In case the amount

[1] `https://www.oracle.com/technetwork/articles/sharma-indexes-093638.html`

of columns matches an exact multiple of the machine word length, there is no fragmentation. In the rare event that a bitmask would extend beyond a single disk block, a linked-list mechanism can be used to chain the blocks containing the mask together.

In practice, however, bitmasks are typically stored in compressed form using some form of lossless encoding such as run-length encoding. In addition to bitmask compression, one can also encode the attribute values, lowering the amount of rows required to store the same amount of entity data. However, it is generally true that encoding sacrifices performance in order to lower the spatial storage requirements. The best compression technique is thus different for each use case as it depends on the hardware of the machine housing the index and the willingness of the system administrator to trade space for performance.

It should be noted that, aside from the bitmap, the database system must also maintain look up tables which map the column indices to attribute values and back, and entity identifiers to the addresses of the disk blocks where the reside. Nevertheless, these are generally insignificant compared to the size of the bit matrix itself.

Hashing techniques

Another popular technique for indexing is based on hash functions. A hash function is essentially a mathematical transformation of any value to a different value in a way that certain properties are satisfied, namely:

- Hashing the same value twice will yield the same result.
- Hashing two similar values will result in two very different values.
- It is rare that two different values will result in the same value.

In the context of indexing, hash functions are used to translate attribute domain values to disk block addresses. This poses a few challenges, as there are a finite amount of disk block addresses, but a possibly infinite amount of domain values.

Hash functions have the advantages of not having any storage requirements (only the hash function itself needs to be stored) and of having extremely good performance (constant order). Indeed, given a specific value, a hash function is capable of finding the disk block containing the target entity data in only one operation, implying that only a single disk block must be loaded. In addition, hash functions do not require that the domain of the indexed attribute is sortable.

Hash functions also suffer from particular drawbacks. For instance, they are very poor at evaluating range queries. Alternatively, it is difficult to construct a hash function that maintains its desirable properties as datasets evolve over time, more precisely when they grow. This has lead to dynamic hashing techniques such as extendible hashing [11] and linear hashing [12]. For hashing values from multiple attributes, a technique called partitioned hashing was proposed, wherein essentially different attribute values are hashed to part of a disk

block address, and the entire disk block address is obtained by concatenating the results of the individual hashes.

5.1.6 Balanced tree-based indices for fuzzy data

2BPT

In 2008, Barranco et al. proposed 2BPT indices for fuzzy numerical data [6]. The index is based on the combined usage of two BPTs, hence the acronym 2BPT. 2BPT treats the lower and upper bounds of the support of the indexed fuzzy numbers as separate attributes. Rather than using a CBPT, two separate BPTs are used.

Inserting an interval in a 2BPT index corresponds to inserting its upper bound in one BPT index and its lower bound in another BPT index. Computing the preselection set for a query with convex membership function P, whose support corresponds to the interval $[P_a, P_d]$, happens in three steps:

1. Perform the look-up of P_a in the BPT on upper bounds to find the fuzzy numerical value whose uppermost possible value is closest to P_a. Assuming the upper bounds are sorted in natural order (ascending), following the P_{next} pointers of the leaf nodes leads to the set of intervals T_1 for which Eq. 5.4 does not hold, i.e. for which the (uncertain) datum is certain to end before P_a.

2. Perform the look-up of P_d in the BPT on lower bounds to find the fuzzy numerical value whose lowermost possible value is closest to P_d. Assuming the lower bounds are sorted in reversed natural order (descending), following the P_{next} pointers of the leaf nodes leads to the set of intervals T_2 for which Eq. 5.3 does not hold, i.e. for which the (uncertain) datum is certain to start after P_d.

3. The intersection $T = T_1 \cap T_2$ describes the intervals for which neither equation holds. In other words, T is the preselection set.

Note that, even though the first two steps can be executed in parallel, the last step must wait until both steps are completed. Alternatively, the third step can be performed during the execution of the second step, but in that case, the first two steps must be executed sequentially. In any case, the evaluation comes down to a procedure where different steps must be executed sequentially and possibly the system must thus wait for a previous step to complete.

Note that 2BPT must furthermore traverse both linked lists of leaf nodes of the individual indices entirely, from the found fuzzy numerical value until the end, in order to compute the preselection set. Since visiting a leaf node corresponds to the costly operation of reading a disk block, this has two disadvantages:

1. The performance of 2BPT is linearly correlated to the amount of leaf nodes and thus the size of the data set.

2. 2BPT will unavoidably visit leaf nodes that are not part of the intersection, i.e. disk blocks that do not contribute to the preselection set.

As indices should try to minimize loading irrelevant disk blocks, this signals a possible area to improve upon 2BPT.

2ABPT

In [14], Medina et al. propose a variant of 2BPT called 2ABPT, wherein the two component BPT indices of 2BPT are replaced by CBPT indices: one on upper bounds first and one on lower bounds first. The advantage is that having all interval information in each tree makes it possible to compute the preselection set by traversing only one of the CBPTs, rather than having to traverse both BPTs. Either CBPT index can be used to compute the full result, but one will always be more efficient than the other, depending on the query and the characteristics of the data set. Their experiments show that the query analyser is well capable of selecting the most efficient one based on its empirical knowledge of the data set. The largest remaining downsides of 2ABPT are that such an index is rather large and that it requires the simultaneous upkeep of two separate CBPT indices in order to keep them synchronized.

5.2 Preselection

Constructing a suitability distribution to represent the degree to which a fuzzy datum satisfies a fuzzy query is a costly operation, involving the analysis of the possibility distribution denoting the fuzzy datum and the fuzzy set denoting the user preferences. In some cases, however, this evaluation can be avoided. While studying the subject, Bosc et al. were the first to notice that it is possible to easily identify the records for which it is certain that they can not satisfy the query [9] in case both the fuzzy value and the fuzzy query have a convex membership function. In a database querying context, this implies that those records do not have to be fetched and processed in order to construct the query response. The so-called preselection principle observes that if the support of the fuzzy value does not overlap the support of the fuzzy query, the value is guaranteed not to satisfy the query.

Consider the criterion a IS c, where a is an attribute of a relation in a relational database supporting uncertainty and c is a fuzzy set expressing the user preferences regarding the desired values of a. Assume criterion c is a trapezoidal fuzzy set that is fully defined by the quadruple $(\mathrm{c}_a, \mathrm{c}_b, \mathrm{c}_c, \mathrm{c}_d)$. Testing if the record r whose value for a is given by another trapezoidal fuzzy set $\pi_a(r) = (\pi_a^r, \pi_b^r, \pi_c^r, \pi_d^r)$, may be rejected, requires two checks:

$$\pi_a^r > \mathrm{c}_d \tag{5.3}$$

and

$$\pi_d^r < \mathrm{c}_a \tag{5.4}$$

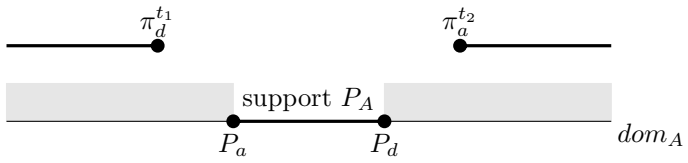


Figure 5.9: Visualisation of preselection for processing query criterion A IS P. Rejection areas for P are highlighted in grey. Based on Eq. 5.3 (resp. 5.4), record t_2 (resp. t_1) is rejected.

Indeed, if either Eq.5.3 or Eq.5.4 is true, the supports of $\pi_a(r)$ and c do not overlap and r can never satisfy c and thus can be safely rejected. This is visualized in Figure 5.9.

The preselection principle hence provides two simple mathematical tests that can be used in order to determine which records should *not* be further processed. This is useful because it allows us to identify entities that can be rejected from the result set without having to compute the suitability distribution first. In fact, this is true even when suitability distributions are not used, too. Not only does preselection lower query evaluation time, it also offers a large degree of freedom when it comes to choosing which evaluation technique to use, as the actual evaluation can be postponed until after preselection is done.

The idea of preselection essentially comes down to *interval boundary comparison* and has sparked the invention of several indexing techniques for interval data (and by extension for possibilistic data). Bosc et al. first suggested using an index built by superimposed coding to annotate the support and core of a possibility distribution [10]. Yazici et al. proposed to use a single, multidimensional index for similarity-based fuzzy data in [18, 19]. Their technique can be applied for attributes whose domain consists of a finite amount of fuzzy sets that are denoted by linguistic labels. Liu et al. presented 1GT, a technique based on G-trees that can be used for convex possibility distributions [13]. In [6, 3, 2] Barranco et al. proposed 2BPT. This technique employs two B^+-trees: one for the lower and one for the upper bounds of the supports of the indexed (convex) trapezoidal possibility distributions. Since this later work, there are proposals index structures for scalar fuzzy scalar data [8, 7], and designed to solve necessity based queries [4, 5]. Also, some works make wide evaluation of indexes proposal in the possibility [14, 16] and necessity contexts [15, 14].

5.3 Interval B^+-trees

In this section, we propose a new technique for indexing interval data called an interval B^+-tree (IBPT). An IBPT is essentially a composite BPT with an extra pointer list connecting the leaf nodes in order to optimize query evaluation for interval data. More precisely, an IBPT index is a two-column index on both bounds of the indexed intervals. It has already been shown that a single

composite BPT index can be used to compute the preselection set, but the procedure to do so requires that all leaf nodes are traversed using the P_{next} pointer list, which can be inefficient in many cases. Medina et al. [14] mitigated this problem by creating an additional composite BPT and letting the query optimiser decide which index to use in a super index they call 2ABPT. IBPT solves the problem by adding an additional pointer to each leaf node which form an alternative linked list of all leaf nodes, enabling a traversal of all leaf nodes in a different order than the one imposed by the P_{next} pointers.

Just like is true for any CBPT, the order in which the attributes are specified matters. For a certain attribute, one can build two different IBPTs: one on lower bounds first or one on upper bounds first. As will be shown in the experiments, this choice has an impact on query performance. For the remainder of the chapter, we assume that the specified order is upper bound, lower bound. We will use the symbol d to represent key values in internal nodes. These represent the upper bounds of the indexed intervals.

5.3.1 IBPT Structure

For an IBPT index of order b, an IBPT internal node contains at most $b-1$ keys, $d_i, i = 1, \ldots, b-1, d_1 < d_2 < \cdots < d_{b-1}$, and b tree pointers $P_i, i = 1, \ldots, b$ to b subtrees $S_i, i = 1, \ldots, b$ such that each key d in S_i satisfies $d_{i-1} \leq d < d_i$. It is easy to verify that IBPT internal nodes are structurally identical to BPT internal nodes.

An IBPT leaf node contains three pointers to other leaf nodes (P_{next}, P_{prev} and P'_{prev}) and at most q intervals $(a_i, d_i), i = 1, \ldots, q$ with their corresponding data pointers $D_i, i = 1, \ldots, q$. Herewith, $q \leq l$, with l being the largest natural number for which

$$l(||P|| + 2||k||) + 3||P|| \leq ||B|| \tag{5.5}$$

holds. Roughly, $l = 2b/3$. The P_{next} pointer is identical to the one of a regular BPT leaf node such that following the P_{next} pointers corresponds to iterating over all indexed intervals according to the chosen order of the chosen bounds (here: increasing upper bounds).

The novel aspect of IBPT is the introduction of a secondary, doubly linked list that connects all leaf nodes in a different order. The idea behind this secondary linked list is to have a data structure that additionally provides sorted access to the intervals according to the other bound (here: lower bounds), in both decreasing and increasing order. If there were exactly one interval per leaf node, it would be possible to create this list perfectly. However, because a leaf node contains multiple intervals, grouped and sorted by their upper bounds, only an approximate sorting is achievable. For any leaf node l, let $\underline{a}(l)$ be the minimal lower bound of all intervals in l. The leaf nodes can be ordered according to their minimal lower bound. In order to realize the secondary linked list, two extra leaf node pointers (respectively P_{prev} and P'_{prev}) must be added to each leaf node. Though only P_{prev} is necessary to compute the preselection set, the P'_{prev} pointers significantly lower the cost of insert and delete operations.

Before we continue, let us briefly reflect on the structural impact of adding additional pointers to each leaf node. Adding the proposed two additional pointers to each leaf node means that, given that the size of a leaf node is fixed, leaf nodes can store less key values, which in turn has an influence on the size of the tree. How big this impact is depends on how large the two pointers are compared to the size of a leaf node. In practice, this is typically very small: a modern hard disk block is generally 4096 bytes large, a pointer is 8 bytes. The ratio of two pointers to a leaf node is roughly 0.4%, which is practically negligable.

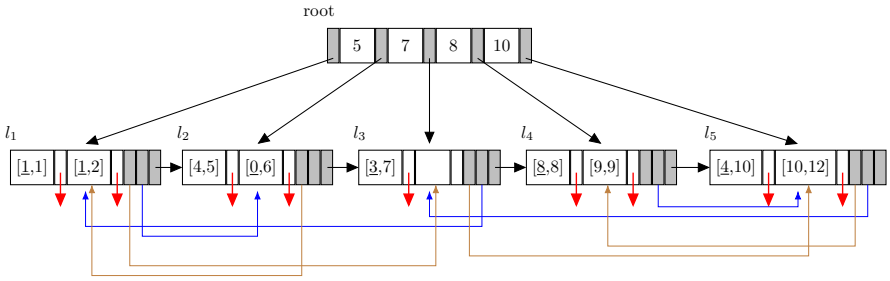


Figure 5.10: Example of an upper-bound based IBPT of order 5: internal nodes can store 4 values, leaf nodes can store 2 intervals. The minimal lower bound is underlined for each leaf. The following intervals were inserted (in order): [8,8], [4,5], [1,1], [3,7], [1,2], [10,12], [4,10], [0,6] and [9,9]. The secondary linked list, indicated by the blue arrows, order the leaf nodes by $\underline{a}$, yielding $l_4 \leftrightarrow l_5 \leftrightarrow l_3 \leftrightarrow l_1 \leftrightarrow l_2$ (left to right indicates P_{prev}, as shown in blue, and right to left indicates P'_{prev}, as shown in brown).

Figure 5.10 shows an example of an upper bound based IBPT of order 5. The same keys that were used in the example from Figure 5.8 are now considered as upper bounds for random intervals and are inserted in the same order.

5.3.2 Impact on Insert and Delete Procedures

Each time the index is changed (i.e. when an interval is added or deleted), the correctness of the secondary pointer list must be validated. If no longer valid, the lists must be repaired. In the following, we will cover the impact on the performance of insert and delete operations.

Insertion. Assume we are adding the interval $[a_n, d_n]$ to an existing IBPT index. The process to do so is as follows. First, the appropriate leaf node where the interval should be inserted is determined by applying the insert algorithm of regular BPTs using d_n as a look-up value. Let us call this leaf node l_c. After inserting the interval in l_c, two situations can occur: either l_c is overflowing and

must split, or l_c is not full yet and the interval has been successfully inserted. Let us first consider the situation where l_c is not full yet. In this situation, again two cases can appear: either $a_n < \underline{a}(l_c)$ or $a_n \geq \underline{a}(l_c)$. In case of the latter, the insert operation is finished and the secondary list does not need to be updated. In case of the former, however, $\underline{a}(l_c)$ changes to a_n and the secondary linked list must be evaluated to check if it needs to be repaired. If $a_n < \underline{a}(P_{prev})$, l_c is no longer in the correct position in the list and the secondary linked list must be repaired. In order to do so, one must follow the P_{prev} pointers until a leaf node l is reached such that $\underline{a}(l) \leq a_n$. When this leaf node is found, let $l_p = P'_{prev}(l_c)$, $l_n = P_{prev}(l_c)$ and $l_m = P'_{prev}(l)$. In order to repair the linked list, these pointers must then be updated in the following order:

1. $P'_{prev}(l_n) \leftarrow l_p$
2. $P_{prev}(l_p) \leftarrow l_n$
3. $P_{prev}(l_c) \leftarrow l$
4. $P'_{prev}(l) \leftarrow l_c$
5. $P_{prev}(l_m) \leftarrow l_c$
6. $P'_{prev}(l_c) \leftarrow l_m$

Example. Let us insert the interval $[0.5, 7.5]$ in the example of Figure 5.10. The interval would be inserted in l_3 and because $0.5 < 3$, $\underline{a}(l_3)$ changes and the secondary pointer list must be updated. Following $P_{prev}(l_3)$ brings us to l_1. Because $\underline{a}(l_1) = 1 > 0.5$, we must continue traversing the secondary list using the P_{prev} pointers. $P_{prev}(l_1)$ brings us to l_2. This time $\underline{a}(l_2) = 0 \leq 0.5$, so we have found the right place to insert l_3. The following pointers are updated:

1. $P'_{prev}(l_1) = l_5$
2. $P_{prev}(l_5) = l_1$
3. $P'_{prev}(l_3) = l_2$
4. $P'_{prev}(l_2) = l_3$
5. $P_{prev}(l_1) = l_3$
6. $P'_{prev}(l_3) = l_1$

The secondary, doubly-linked list now corresponds to $l_4 \leftrightarrow l_5 \leftrightarrow l_1 \leftrightarrow l_3 \leftrightarrow l_2$ (descending lower bounds from left to right).

Let us now consider the case where l_c is full and overflows. Again, let $l_n = P_{prev}(l_c)$ and $l_p = P'_{prev}(l_c)$. Before splitting the node, the secondary list has to be updated so that $P'_{prev}(l_n) = l_p$ and $P_{prev}(l_p) = l_n$ (in words: l_c is removed from the secondary list). Next, the regular splitting strategy is applied and the intervals are distributed among the two new leaf nodes based on their

upper bounds. After redistributing the intervals across the two new leaf nodes, it is almost guaranteed that one of them will have to move "upstream". The other can either stay in place or has to be inserted "downstream", similar to the case when l_c did not overflow. The reinsertion procedure for these two new leaf nodes is similar to the one described above but the pointers used to traverse the secondary list depend on whether the node should be inserted up- or downstream. To insert the new leaf l (with minimal lower bound $\underline{a}(l)$), P'_{prev} is used if $\underline{a}(l) > \underline{a}(l_n)$ (upstream) and P_{prev} is used if $\underline{a}(l) < \underline{a}(l_n)$ (downstream).

Clearly, the complexity of inserting an interval in an IBPT is higher than the complexity of the insert procedure of a regular BPT. Each interval that is inserted can trigger an update procedure which could, in the worst case, cause the traversal of all leaf nodes. IBPT should be preferred in situations where the data is queried (much) more frequently than it is modified. For batch insert operations, it is advised to create the IBPT index without maintaining the secondary pointer list and to postpone the construction of the secondary linked list until the entire batch has been inserted.

Deletion. Similar to insertion, each deletion might invalidate the secondary list. Assume that the interval $[a_n, d_n]$ has to be deleted from an existing IBPT index. This can be achieved by applying the following procedure. First, the look-up algorithm is applied using d_n in order to find the leaf node l_c which contains $[a_n, d_n]$. Should the index not contain this interval, the procedure ends without making any modifications. If it does, $[a_n, d_n]$ is removed. Two possible situations arise: either l_c is underflowing, or it is not. The simplest case to handle is that in which it is not underflowing. In that case, $\underline{a}(l_c)$ is computed (after the removal of $[a_n, d_n]$). If $\underline{a}(l_c) \leq a_n$, then removing a_n did not have an impact on the secondary linked list, as it was clearly not the smallest lower bound in l_c, and the deletion procedure is finished. Otherwise, $\underline{a}(l_c) > a_n$ (a_n was the minimal lower bound in l_c), then the secondary linked list must be validated as the order may no longer be correct: indeed, given that $\underline{a}(l_c)$ has increased, it is possible that l_c must be moved "up" the secondary linked list. The P'_{prev} pointers can be used to find the first leaf node l for which $\underline{a}(l) \geq \underline{a}(l_c)$. If no such node exists (the end of the list is reached), l_c must be moved to the head of the secondary linked list. In all other cases, l_c is removed from the secondary list and re-inserted just before l, such that $P'_{prev}(l_c) = l$ and $P_{prev}(l) = l_c$.

Let us now consider what happens if l_c underflows after removing $[a_n, d_n]$. In this case, the regular BPT algorithm for dealing with underflow is applied (as described in section 5.1.4). During this process, either l_c steals a value from one of its neighbors, or it merges with one entirely, absorbing a value from its parent (possibly propagating all the way up to the root). If a value was stolen, both l_c and that neighbor must be deleted from the secondary list and re-inserted in their appropriate places. If l_c and a neighbor merged, l_c and that neighbor are removed from the secondary list and the resulting single leaf is re-inserted.

Example. Let us delete the interval $[3, 7]$ from the example index shown in Figure 5.10. Looking up the value places us in leaf node l_3. By removing $[3, 7]$, l_3 is underflowing. According to the underflow procedure, first the "left" neighbor, l_2, is consulted. As it happens, l_2 contains sufficient entries so that it can donate one and still not be underflowing. As such, it donates $[0, 6]$ to l_3. Because no merge procedure has occured, both l_2 and l_3 must be removed from and then re-inserted into the secondary list. The resulting tree is shown in Figure 5.11. It shows that l_3 has been moved downsteam to take over the position of l_2, as it received $[0, 6]$, the interval that was deciding for $\underline{a}(l_2)$. Meanwhile, l_2 has moved upstream because its minimal lower bound has increased from 0 to 4, placing it just after l_5 in the P_{prev} order. The secondary, doubly-linked list now corresponds to $l_4 \leftrightarrow l_5 \leftrightarrow l_2 \leftrightarrow l_1 \leftrightarrow l_3$ (descending lower bounds from left to right).

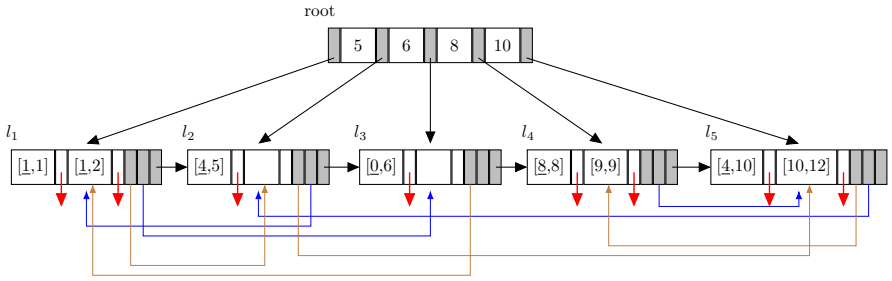


Figure 5.11: Example of an upper-bound based IBPT built using [8,8], [4,5], [1,1], [3,7], [1,2], [10,12], [4,10], [0,6] and [9,9], after removing [3,7].

5.3.3 Applying Preselection

It has already been established in 5.1.4 that the preselection set can be constructed by traversing all the leaf nodes of a composite index on both lower and upper bounds. In what follows, we will describe a procedure that allows us to construct the preselection set by traversing fewer than all leaf nodes. The procedure hinges strongly on the following two observations:

- to exploit Eq. 5.3, we can use the P_{prev} pointers, and
- to exploit Eq. 5.4, we can use the P_{next} pointers.

Assume we have to evaluate a fuzzy query of the form a ISc, where c denotes preferred values for attribute a modelled by a convex membership function whose support is bounded by $[\mathrm{c}_a, \mathrm{c}_d]$ (we will use this notation, for the sake of clarity). We will now show how an IBPT index can be used to compute the preselection set. Starting the procedure is done by initializing three variables:

- $S = \{\}$, a set that will be used to collect intervals that match the query.

- $M = P_{next}$, a traversal method that is used to determine how to navigate the leaf nodes.
- $E = \{f_1(a,d) = I_{d<\mathrm{c}_a}, f_2(a,d) = I_{a>\mathrm{c}_d}\}$ (where $I_A(x)$ is the indicator function, returning 1 if x satisfies A and 0 otherwise), a set of functions that is used to evaluate the preselection predicates on intervals $[a,d]$.

Initially, S is empty, M dictates the usage of the primary linked list through P_{next} pointers and E contains prediates that test both Eqs. 5.3 and 5.4. After applying the following procedure, S should contain the preselection set:

1. Perform the look-up algorithm for the value c_a in order to obtain leaf node l.
2. Test all intervals in l and add only those that satisfy none of the predicates in E to S. Meanwhile, keep a running minimum of the lower bounds so as to efficiently compute $\underline{\mathrm{a}}(l)$.
3. If $\underline{\mathrm{a}}(l) \leq \mathrm{c}_d$, remove f_1 from E, load the next leaf (as indicated by M) and go back to the previous step. Otherwise, move to the next step.
4. Determine (e.g. using a heuristic) whether or not it is beneficial to change M and then proceed to the final step. Consider, for instance, the following heuristic. Consult the root node and determine the index i of the subtree S_i that contains the upper bound of the first interval from the current leaf node. In other words, for upper bound d, find which value for i satisfies $k_i < d \leq k_{i+1}$ for the root node. Assuming that the root node contains s subtrees, check whether $i > s/2$. If it is, do not change M or E. Otherwise, change M to P_{prev} and add f_1 to E.
5. Fetch the next leaf node according to M and add those intervals that satisfy (none of) the predicate(s) in E to S. Repeat this step until the end of the linked list is reached.

After this procedure, S contains the preselection set.

The first step serves as a way to identify the leaf node l from where traversal will start. By applying the regular BPT look-up algorithm with the value c_a, the leaf node that contains the interval whose upper bound is closest to c_a is found. All intervals in this leaf node must be tested using both Eq. 5.4 and Eq. 5.3 because this node might contain intervals that satisfy either. As it is the central theory of ABPT, we know that following the P_{next} pointers will result in finding all the intervals that make up the preselection set. However, if $\underline{\mathrm{a}}(l) > \mathrm{c}_d$, traversing the secondary linked list by following the P_{prev} pointers will also result in finding all the intervals. Therefore, we check to see if $\underline{\mathrm{a}}(l) > \mathrm{c}_d$ and if it is not, we simply load the next leaf node using P_{next} (the default mechanism contained in M which is guaranteed to work for an upper bound based CBPT) and remove f_1 from E. Indeed, by using c_a as a look-up value in an index built (and thus sorted) on upper bounds, it is guaranteed that all intervals in leaf nodes following (according to P_{next} traversal) the leaf node containing c_a (or

its closest indexed upper bound value) can no longer satisfy Eq. 5.4, so f_1 does not need to be tested anymore.

This process continues until a leaf node is reached for which $\underline{\mathrm{a}}(l) > \mathrm{c}_d$, at which point there are two possible ways to proceed: by following the P_{next} pointers or the P_{prev} pointers. The question is: which list is shorter, and will thus find the remaining intervals faster? In order to answer this question, we propose a heuristic that tries to estimate how far along the primary linked list the current leaf node l is. If it is deemed likely that l is over halfway of the primary linked list, then it is assumed that following the P_{next} pointers will be faster than switching to the secondary linked list, and vice versa. The rationale behind the heuristic is as follows. Not much can be said about the amount of leaf nodes that will be visited using the P_{prev} pointers, so it is assumed that, on average, half of the total amount of leaf nodes will be visited until the end of the list is reached. The linked list derived from the P_{next} pointers, however, is well known. When starting from a leaf node in the right half of the tree, using the P_{next} pointers would lead to having to visit *at most* half of the total amount of leaf nodes. Conversely, using the P_{next} pointers in the left half of the index would lead to having to visit *at least* half the total amount of leaf nodes. In such cases, it is decided to use the P_{prev} pointers because, on average, it will require that only half of the leaf nodes be traversed, which is better than at least half. Obviously, guessing which side of the tree the current leaf node is in by using only the index of the subtree containing the first search key value, based purely on the root node is a simple but crude estimate. One could think of a more sophisticated heuristic, especially if additional metadata regarding the secondary linked list is stored, but this would generally come at a trade of between insert/delete performance versus look-up speed. This offers an interesting way for database administrators to influence the performance of IBPT indexing based on their actual use-case.

If the heuristic decides to switch, M is changed to reflect this, and E is updated to contain f_2 instead of f_1. Indeed, when following the P_{prev} pointers, it is no longer guaranteed that all intervals satisfy Eq. 5.4, so f_2 must be re-added to E. Instead, it now becomes a certainty that no remaining intervals can satisfy Eq. 5.3, so f_1 can be removed from E.

Note that in case the heuristic decides not to switch, the procedure is essentially identical to the query evaluation mechanism found in a regular CBPT index. As such, the heuristic plays a role similar to that of the query optimiser in 2ABPT in the sense that it will estimate which option is the fastest way to calculate the preselection set.

5.3.4 Example

We will demonstrate the preselection procedure using an example. We will not yet concern ourselves with counting how many disk blocks are loaded; this is discussed in greater detail later, in section 5.4. This example is purely to show how the algorithm functions.

Let us revisit the example from Figure 5.10. Figure 5.12 is a graphical

representation of the indexed records (sorted by upper bounds, as in the IBPT index) and a user preference (indicated by vertical dashed lines). This particular user preference can be interpreted as a desire for records whose value is between 6 and 7. For a small data set like this one, we can easily identify the preselection set (i.e. the set of intervals that have at least one value in common with $[6, 7]$) on sight. Let us call this set T:

$$T = \{[0, 6], [3, 7], [4, 10]\}$$

Let us now follow our procedure step-by-step, keeping track of the records that are identified as candidates in the set S. Initially, $S = \emptyset$, $M = P_{next}$ and $E = \{f_1(a, d) = I_{(d<\mathrm{c}_a}, f_2(a, d) = I_{a>\mathrm{c}_d}\}$. After the procedure, S should be equal to T.

1. We search the IBPT for the lower bound of the query (i.e., 6) and arrive at leaf node l_2.

2. Testing all intervals in this leaf against Eq. 5.4 and Eq. 5.3 yields:

   - reject $[4, 5]$ (satisfies Eq. 5.4)
   - add $[0, 6]$

   Meanwhile, calculate $\underline{\mathrm{a}}(l_2) = 0$.

3. Because $\underline{\mathrm{a}}(l_2) \leq 7$, load the next node according to P_{next}, remove f_1 from E and repeat the previous step.

4. The next node is l_3. We test all intervals with f_2 and find that $[3, 7]$ is added to S because it does not satisfy f_2. At this point, $S = \{[0, 6], [3, 7]\}$. We find that $\underline{\mathrm{a}}(l_3) = 3$. Because $3 \leq 7$, we again repeat step 2.

5. Still following P_{next} brings us to l_4. We reject both $[8, 8]$ and $[9, 9]$ (both satisfy f_2) and find $\underline{\mathrm{a}}(l_4) = 8$. Because $8 > 7$, we will apply the heuristic in order to determine which list to follow for continuing our traversal.

6. From the root node, we can identify that l_4 is (in) the 4th subtree from its 5 children. As $4 > 5/2$, it is assumed that it is likely fastest to continue using the primary linked list. As such, we do not have to update anything, and traversal proceeds using P_{next}.

7. The next node that is visited this way is l_5. Here, we add $[4, 10]$ and reject $[10, 12]$. Because $P_{next} = \perp$, the procedure ends.

After the procedure, S contains the following records:

$$S = \{[0, 6], [3, 7], [4, 10]\}$$

This confirms that $S = T$. Note that continuing to use the primary linked list based on the heuristic resulted in visiting one more leaf node. If we would have switched to the alternative linked list, however, we would have had to visit four

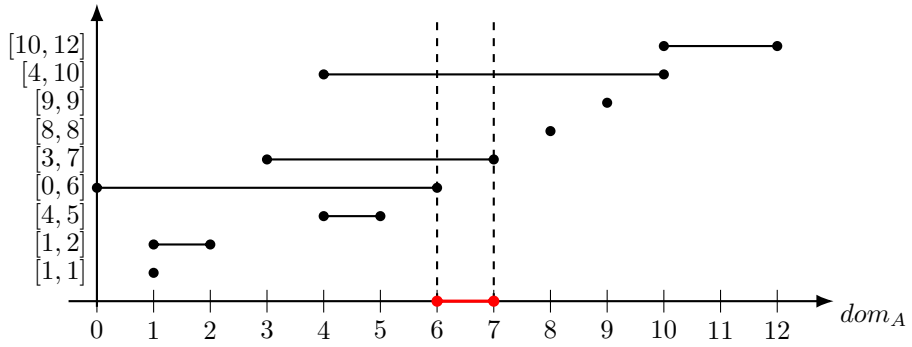


Figure 5.12: Visualization of the indexed records and a query for all intervals overlapping with a value between 6 and 7 (inclusive).

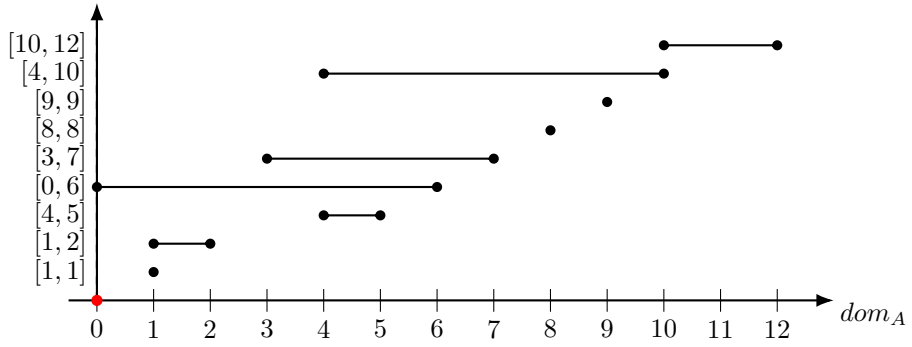


Figure 5.13: Visualization of the indexed records and a query for all intervals overlapping with 0.

more leaf nodes (as indicated by the blue arrows in Figure 5.10), which would have been much less efficient.

Imagine now that we want to execute another query (visualized in Figure 5.13), this time for all intervals that contain 0 (i.e. overlap with the range $[0, 0]$). In this case, we should find:

$$T = \{[0, 6]\}$$

- Search for 0 and arrive at l_1.
- Test $[1, 1]$ and $[1, 2]$ only to reject both and find that $\underline{a}(l_1) = 1$.
- Because $\underline{a}(l_1) > 0$, proceed by applying the heuristic.
- From the root node, this time we find that l_1 is (in) the 1st subtree from the 5 children. Because $1 \leq 5/2$, it is recommended to switch to the alternative linked list. Hence, M is changed to P_{prev} and E is updated to again include f_1.
- Now following P_{prev}, the next leaf node to visit is l_2. The intervals in this leaf node are subjected to the predicates in E, which yields:
  - reject $[4, 5]$ (satisfies Eq. 5.3)
  - add $[0, 6]$ (satisfies neither)

  Because $P_{prev} = \perp$, the procedure ends.

Again, $S = T$. It is purely coincidental that l_2 follows after l_1 in the alternative linked list as well as in the primary linked list. However, this example clearly shows that the heuristic was right to make the switch, because using the primary linked list would have resulted in visiting every leaf node, whereas switching led to only having to visit one more leaf node. Indeed, we know that any potential intervals with a lower bound smaller than 0 can be found by following the P_{prev} pointers, and, thanks to the alternative linked list, we know that no more intervals exists with a smaller bound than $\underline{a}(l_2)$.

5.4 IBPT Analysis

In this section, we perform and report multiple experiments to measure the performance of the IBPT index structure. The experiments use generated, fuzzy data and random, fuzzy queries. For each experiment, we measure the amount of disk blocks that are transferred (DBTs) from the hard disk into rapid access memory by applying the preselection procedure described in Section 5.3.1. Aside from the performance of IBPT, we will also measure the performance of a CBPT index (on upper bounds first), a 2BPT index, a 2ABPT index and the baseline where there is no index.

5.4.1 On the Data Set Generation

For the experiments, we use synthetic, generated data sets. Doing so is in fact beneficial for analyzing the performance of IBPT because it gives us control over various aspects of the data set so we can measure how the characteristics of the data set influence the performance of IBPT.

Before we continue, let us first describe the process that is used to generate data sets. Understanding this process is important, because it has implications regarding the random nature of the generated data. To clarify: the generated data sets are essentially lists of intervals. Among these intervals can be intervals of length zero, i.e. representing a single value rather than a range of values.

The intention was to create a procedure that is capable of generating random data sets, but, as it turns out, *random* is not so easily defined. For simplicity, we have limited ourselves to dealing with bounded, discrete domains. This restriction follows naturally from using computers (which inherently work in a discrete, numerical way) and the observation that many properties have some form of natural bounds. Furthermore, data set generation is controlled with the following four parameters:

- an infimum (integer value),
- a supremum (integer value),
- the desired size of the data set (amount of intervals that should be generated), and
- the maximal fuzziness (i.e. interval length) that is allowed in the data set.

The procedure is essentially a loop in which unique intervals are generated until the desired data set size is reached. Initially, a list of all integer values ranging from the lower bound up to and including the upper bound is generated and shuffled. This list represents all unused bounds in the specified range. In each loop iteration, two random numbers are selected from this list of available bounds and the corresponding interval is proposed. It is possible that the upper and lower bound are the same number, in which case the interval denotes a precise value. It is tested whether or not the length of the interval exceeds the defined maximal interval size. If this is not the case, the interval is emitted and the bounds are removed from the list. This ensures that all lower and upper bounds, across all intervals, are unique. Though this is not necessary for using IBPT, it simplifies the tree structure. If the interval exceeds the defined maximal interval size, the process increases a failed-attempt counter and tries again. The failed-attempt counter has the sole purpose of ensuring the process can not get stuck in an infinite loop.

It is important to note that, using this data generation technique, the lengths of the intervals are *not* uniformly distributed: there is a higher chance to generate an interval with a short length. This is because the closer the arbitrarily chosen lower bound is to the supremum that is used to generate the data set, the smaller its possible intervals are. Thus, the truly long intervals can only be

generated if a small lower bound is chosen, whereas a short interval can be generated for nearly all lower bound values. Therefore, longer intervals are much less likely to be generated. It is possible to use a different approach in order to be able to guarantee that the lengths of the intervals are uniformly distributed, but as a consequence thereof, the bounds of the intervals would no longer be randomly distributed across the domain; instead, the lower (respectively upper) bounds will be more concentrated around the infimum (supremum). We decided to choose for the first option because, in practice, we expect that there would be more small intervals (denoting somewhat known values) than large intervals (denoting very uncertain values), especially in case fully unknown values are omitted from the index.

5.4.2 Index applicability

In order to judge whether or not it is justified to create an index, it should be considered whether or not the index is in fact useful when evaluating queries. In order to do that, we must first compute the amount of DBTs that is required to generate the preselection set. The amount that would be needed in case there is no index serves as a base case. However, it is not straightforward to calculate this required amount because it depends on factors like the spatial locality of data and the size of each record. Spatial locality is an indicator for how scattered records are across disk blocks. To illustrate this, consider records that are r bytes large and that a disk block is b bytes large. Technically, it is possible to store $\lfloor b/r \rfloor$ records in each disk block, in which case spatial locality is maximal. In the (unrealistic) case that such a record would denote only an interval and that all records are stored contiguously, scanning all records in a data set consisting of s records would require $\frac{s}{\lfloor b/r \rfloor}$ DBTs. It can already be seen that the amount of DBTs increases with r (as r grows, the denominator shrinks and the fraction becomes larger). In practice, however, records typically describe entities that consist of multiple attributes and as such, r is usually (much) larger than the size required to store a single interval. Moreover, data is rarely stored contiguously. In the worst case, the records are scattered across disk blocks in such a way that there is only one record per disk block. In that case, it does not matter how large r is (as long as it is smaller than b), and computing the preselection set for a given query in the absence of indices would require s DBTs. Reality is arguably much closer to the worst case than to the best case. This becomes apparent when considering the complexity of maintaining contiguity in the presence of delete and update operations. In any case, the complexity of the procedure is linear in s.

It should be stressed that when an index is used, neither spatial locality nor record size has an impact on the amount of DBTs that is required to construct the preselection set. When using a B^+-tree based index, a high degree of spatial locality is guaranteed by design. These indices explicitly focus on organising disk blocks such that the interval data are not only grouped (as to maximize spatial locality) but also sorted (in order to facilitate logarithmic searching). By only copying the values of the indexed attribute(s) (e.g. the bounds of the indexed

intervals), the index is already significantly smaller than the entire data set and exhaustively scanning it would be almost as efficient as the base case. The only reason that the index structure can never be as efficient as the absolute best base case scenario, is because each interval has to be annotated with a pointer to the disk block where the entire record is stored, which implies that storing an interval requires more bits than those needed to describe its bounds. As such, the maximal spatial locality can never be as good as when no index is used (where no additional pointer must be stored per interval). Let l denote the amount of records whose interval datum can be stored in an IBPT leaf node, then iterating over all leaf nodes of such an index requires an amount of DBTs that is linear in s/l. Though this is also linear in s, the factor $1/l$ is guaranteed, even in the absolute worse case where each record is potentially very large and all records are scattered across different disk blocks. It is worth noting that these are exactly the situations in which it is desirable to lower the required amount of DBTs.

All in all, it is possible to give a pretty accurate number predicting the amount of DBTs that will be required to construct the preselection set given an index. Clearly, this is not the case when no index is used, making it hard to say whether or not an index is worth creating. Whether or not an index for a certain data set will be useful depends on the size of the records and their fragmentation across disk blocks. It is safe to say that, unless for read-only databases, fragmentation will accumulate over time up to the point where non-indexed data access becomes infeasible. This means that the same index for the same data set might not be used on machine A, while it is used for that same index for that same data set on machine B. Only the query analyser, which is a part of the database system that has insight into how data is actually stored in that particular instance, is truly capable of estimating whether or not using an index to answer a query is worth it. However, it is not hard to defend that when an attribute is frequently used in queries and the query analyser suggests that using an index is at least sometimes worth it, that the index should be made, as it clearly serves a purpose to speed up what would otherwise be slow queries. Other than that, it is also somewhat of a subjective issue, depending on the requirements (will the data be manipulated often, how important is the speed of select queries versus that of data manipulation operations, …). After this discussion, we find it safe to say that it is at least interesting to try out IBPT in order to see if the query analyser would decide to use it. We know for a fact that it will, in some cases, drastically speed up query evaluation time, and in any case enable us to make very stable and accurate predictions about query evaluation time, which would not be possible without the index. This stability and predictability is in itself valuable, too. But as the experiments point out, IBPT does not just offer stability, it can significantly speed up query evaluation, too.

5.4.3 Influence of the Fuzziness of the Query

Let us now take a look at the influence of some of the characteristics of the data on the efficiency of IBPT. We will first cover the "fuzziness" of the query, by which we mean the amount of acceptable values it specifies. The fuzzier the query, the more values it accepts. In more formal wording: by fuzziness of the query $[P_a, P_d]$, we mean $P_d - P_a$.

The data sets that were generated for these experiments consist of 1000 unique intervals (each with a unique lower and upper bound) in the domain $[b_l, b_u]$ with a maximal fuzziness of $(b_u - b_l)/4$.

Given a discrete domain bound by b_l and b_u. Let $b_\alpha = (1-\alpha)b_l + \alpha b_u$. To measure the influence of query fuzziness, we consider three unique queries:

1. $[b_{4/16}, b_{5/16}]$ (a "small" query),
2. $[b_{1/4}, b_{1/2}]$ (a "median" query), and
3. $[b_{1/4}, b_{3/4}]$ (a "large" query).

Instead of randomizing the position of the query (by choosing a random lower bound, for example), the same queries are repeatedly executed on 100 different data sets. In doing so, we attempt to minimize the influence of the position of the query, which is discussed in the next set of experiments.

The results of the experiments are summarized graphically in Figure 5.14. A first observation is that the amount of DBTs increases with the fuzziness of the query. This is logical, because (given a fixed data set) the more fuzzy the query, the larger the preselection set. A second observation is that the performance of IBPT varies between that of CBPT and 2ABPT. Recall that 2ABPT is essentially a combination of two CBPT indices, one on lower and one on upper bounds, from whom the query analyser picks the most efficient one at query execution time. In cases when the query analyser decides to use the index on upper bounds, it is not surprising that the performance of CBPT is identical to 2ABPT. In all other cases, IBPT performs better than CBPT. Any performance gained can be explained by the extra pointers which enable the secondary traversal order of the leaf nodes. Compared to 2ABPT, the observed lower performance can be explained by the fact that the secondary linked list is inherently less efficient than the primary linked list (as is used in 2ABPT). Our final observation regarding this experiment is that it seems that the largest performance gains can be found in the cases where the query is "small". However, we are not yet in a position to say that there is a causality behind this observed correlation. In the following, we consider multiple small queries and manipulate other parameters to gain a better understanding.

5.4.4 Influence of the Position of the Query

For queries of limited fuzziness (i.e. $P_d - P_a$ is small compared to $b_u - b_l$), we now test the influence of the position of the query on the amount of DBTs required to construct the preselection set. By position of the query we mean an indicator

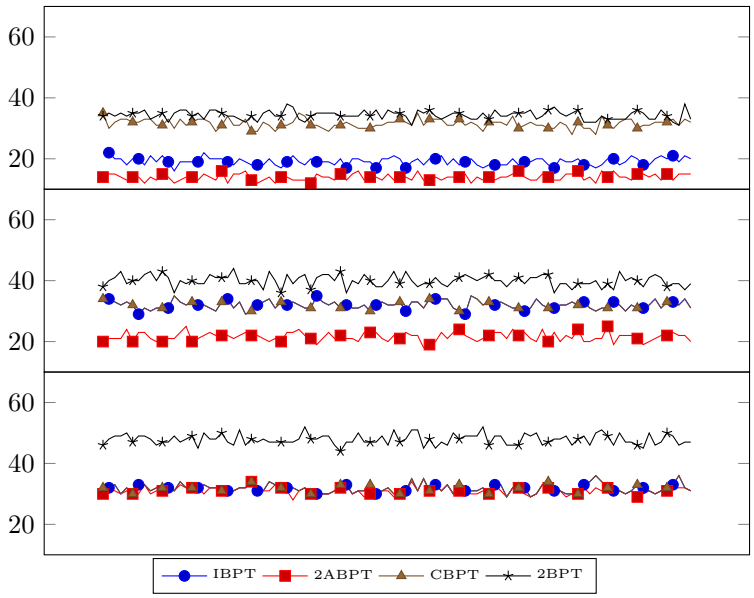


Figure 5.14: Visual representation of the amount of DBTs required to compute the preselection set for varying query fuzziness. From top to bottom, the fuzziness of the query is increased. The x-axis denotes different data sets and the y-axis denotes the amount of DBTs.

of whether or not the interval it denotes predominantly overlaps with the lower half of the domain of the data set or with the upper half. More formally, if $\gamma = (P_a + P_d - 2b_l)/[2(b_u - b_l)]$, then we say that the query prefers low values if $\gamma < 1/3$, the query lies "in the middle" if $1/3 \leq \gamma \leq 2/3$ and the query prefers high values if $2/3 < \gamma$.

The data sets that were generated for these experiments each consist of 1000 unique intervals (each with a unique lower and upper bound) in the domain $[b_l, b_u]$ with a maximal fuzziness of $(b_u - b_l)/4$. To measure the impact of the position of the query, we consider the three following queries:

1. $[b_{3/16}, b_{5/16}]$ (a query preferring low values),
2. $[b_{7/16}, b_{9/16}]$ (a query "in the middle"), and
3. $[b_{11/16}, b_{13/16}]$ (a query preferring high values).

The results of this experiment are depicted in Figure 5.15. From these results it is clear that for queries preferring high values, IBPT performs very similar to 2ABPT and CBPT. This is not surprising, because in these cases, the heuristic of step 3 of the preselection procedure will consistently decide to use step 4a, in which case the procedure for IBPT *is* identical to that of CBPT. Because the queries are small, this seems to correspond to the logic the query analyser uses for 2ABPT, too. However, the lower the values preferred by the query are, the more likely it becomes that the heuristic decides to use step 4b. It is no surprise that in those cases the performance of 2ABPT is better than IBPT (again due to the difference in efficiency of the secondary linked list versus the primary linked list connecting the leaf nodes), but it clearly also outperforms CBPT by avoiding to traverse all leaf nodes. This is interesting, because here we can observe that IBPT can compete with 2ABPT regarding performance, while only taking up as much space as CBPT. Indeed, IBPT combines almost the best of both worlds, providing near 2ABPT performance at a significantly smaller space requirement.

5.4.5 Influence of the Data Set Fuzziness

Next, we will measure the impact of the average fuzziness of the data itself on the amount of DBTs required to compute the preselection set. For these experiments, we consider three sets of parameters to generate data sets, each time limiting the maximal fuzziness of indexed data. Though we will discuss this in greater detail in section 5.4.6, we can already mention here that it is a viable strategy to omit entities whose value is unknown (denoted by the maximal interval to indicate all values are possible) as these overlap with all possible queries and should always be returned. Hence, it makes sense to consider situations where data fuzziness is limited to a certain degree (the "fuzziness threshold"). The special case where there is no uncertainty is also considered. The following three different fuzziness thresholds are used to each generate 100 random data sets of 1000 intervals in $[b_l, b_u]$:

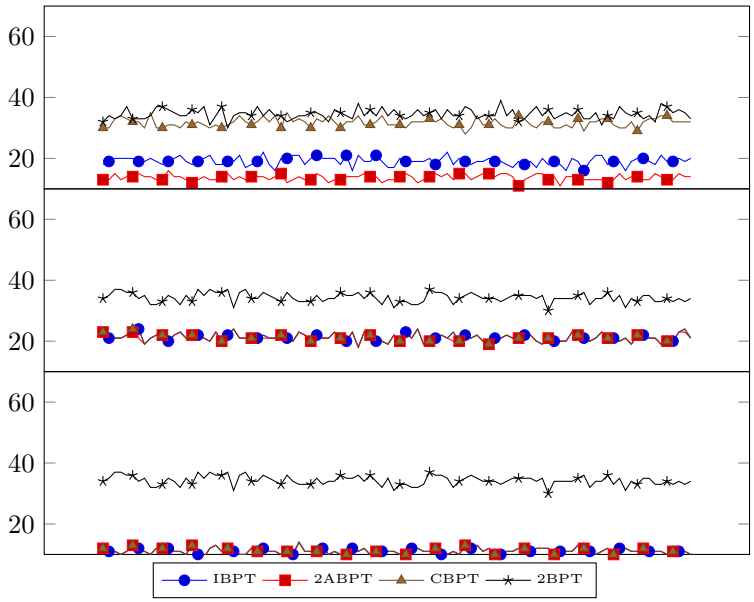


Figure 5.15: Visual representation of the amount of DBTs required to compute the preselection set for varying query position. From top to bottom, the query moves from the left to the right. The x-axis denotes different data sets and the y-axis denotes the amount of DBTs.

1. 0 (all data is certain),
2. $(b_u - b_l)/2$, and
3. $b_u - b_l$ (fully unknown data is allowed and indexed).

The amount of DBTs required to compute the preselection set for the fixed query $[b_{1/8}, b_{3/8}]$ is measured and visualised in Figure 5.16. We have chosen this query because it is a small query that prefers low values, both factors for which we have so far found that they guarantee different behaviour between IBPT and CBPT.

The experiments show that the performance of IBPT decreases as the fuzziness of the data set increases. Obviously, the more fuzzy the data is, the larger the preselection set for any given query will be, but more importantly, the relative performance of IBPT decreases faster with increasing fuzziness compared to other indexing techniques. The reason for this is once again found in the efficiency of the secondary linked list. This efficiency namely depends on the fuzziness of the data. If there is no uncertainty, IBPT has a very similar performance as 2ABPT. This makes sense because the order found by sorting on upper or lower bound is the same as the intervals actually denote numbers and we can sort them perfectly. In such a scenario, the secondary linked list is maximally efficient, hence explaining its high performance. If, on the other hand, fully unknown data are allowed in the index, IBPT will perform poorly because the secondary linked list is almost arbitary and could potentially lead to visiting more leaf nodes when using it compared to simply sticking with the primary linked list connecting the leaf nodes.

5.4.6 Discussion

In this section, we reflect on what the experiments have taught us about IBPT.

Clearly, the secondary linked list (which provides an alternative traversal option for the leaf nodes) strongly impacts the performance of IBPT. The efficiency of the secondary linked list is largely determined by how different its implied order is from the order that would be obtained by actually sorting the intervals by their lower bounds. This, in turn, depends on how similar the order according to their upper bounds is, and how many intervals are clustered per leaf node. As we have already shown, in case there would be no uncertainty (and the order on upper and lower bounds is identical), the secondary linked list is extremely efficient. However, the fuzzier the data, and the more intervals per node, the lower the efficiency. Say that there are, on average, n intervals per leaf node, and that, starting from some arbitrary leaf node l_c, traversing the leaf nodes using the P_{prev} pointers until the end of the secondary linked list would result in visiting t leaf nodes. In the process, roughly nt intervals would be tested, though it might be so that there is only one interval per node for which its lower bound is actually smaller than the minimal lower bound of l_c. That means that, in the worst case, only t intervals from the nt that are tested

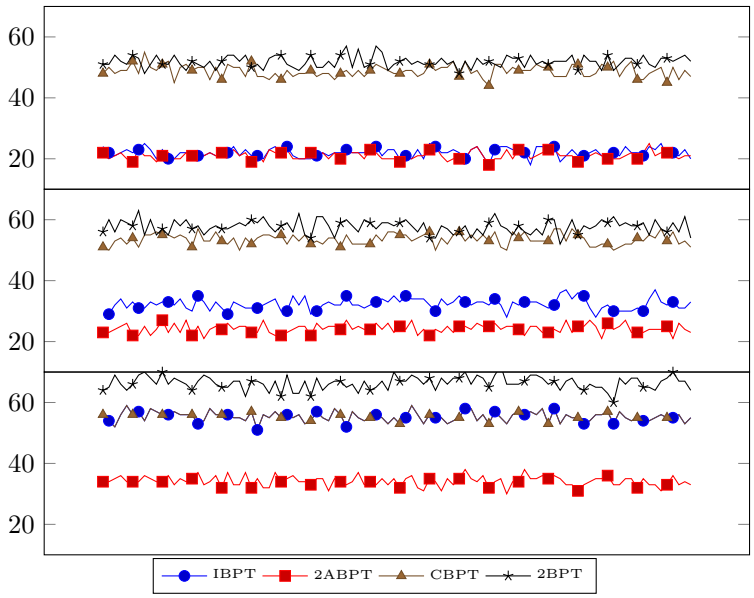


Figure 5.16: Visual representation of the amount of DBTs required to compute the preselection set for varying data set fuzziness. From top to bottom, the fuzziness threshold is increased. The x-axis denotes different data sets and the y-axis denotes the amount of DBTs.

are actually relevant, an efficiency of 1 in n. Conclusively, the larger n (the amount of intervals per leaf node), the lower the efficiency of the secondary list.

Let us now study the properties of the secondary linked list. More precisely, let us try to understand when a P_{prev} pointer will move "right" (i.e. in the same direction of P_{next}) and when it will move "left" (i.e. in the opposite direction of P_{next}). The reason we will investigate this is because we can say a few things about the efficiency of the traversal. Consider that we are evaluating a query and that we start by looking up the lower bound of the query in an upper-bound based IBPT. Due to the nature of tree-based indexing structures, we know that we can immediately exclude all nodes to the left of the first leaf node that is visited because the intervals they contain all end before the lower bound of the range query. This means that we are certain that, by repeatedly following P_{next} and exclusively moving right one node at a time, we will be able to construct the entire preselection set correctly. However, as is proven by the performance of CBPT, it is highly likely that, during this process, we will visit leaf nodes that do not contain any relevant interval. The secondary linked list is supposed to improve the traversal over the leaf nodes by offering us a way to "skip over" such leaf nodes. Essentially, if the P_{next} pointers would have us traverse k nodes, we want to visit less than k nodes by following the P_{prev} pointers. Therefore, it the P_{prev} pointers would have to move to the right, sometimes jumping across nodes. In order for this to happen, there must exist a leaf node, to the right of the current node (i.e. there must exist intervals with larger upper bounds), that has a smaller minimal lower bound than the minimal lower bound of the current node. In other words, there must exist a more uncertain interval than the most uncertain interval in the current node.

Alternatively, the P_{prev} pointers can also move to the left. Their tendency to do so is determined by how well the lower and upper bounds are positively correlated. An extreme case thereof is when there is no uncertainty in the data. If this is the case, the intervals denote precise values (for which a unique partial order exists) and the P_{prev} pointers will coincide with the P_{next} pointers, but in the opposite direction. This property does not only hold in the absence of uncertainty, indeed, it can be seen that this property will keep holding as long as the uncertainty increases equivalently across all entities, as long as the order based on upper and lower bounds remains identical.

As such, it can be concluded that the properties of the P_{prev} pointers depend on the *distribution* of the *uncertainty* across all intervals in the data set. It is plausible to assume that there will always be entities whose value is unknown, but it can be justified to omit them from the index because they will always be part of the preselection set.

We have yet to explain the impact of moving left using the P_{prev} pointers. We already know that it is guaranteed that all entities that should be included in the preselection set regarding a particular query can be found by only moving to the right. As described above, the purpose of the secondary linked list is to allow us to traverse the leaf nodes faster by allowing us to skip over nodes that do not contain any relevant intervals. However, we just mentioned that using the P_{prev} pointers can cause us to move to the left. Moving to the left will

either force us to revisit a node that we have already visited (during step 2) or to visit a node that we have already rejected (in step 1). Revisiting a node that has already been visited is not that big of a problem, because we do not need to load the corresponding disk block into memory again (assuming that the database buffers are sufficiently large). Otherwise, moving left to a leaf node that had not yet been visited essentially implies that we must transfer a disk block of which we know that it can not contain any results for the preselection set. Such an "empty visit" can not be avoided, because we need to extract the next P_{prev} pointer in order to be able to continue the traversal, in the hopes that it might move us to the right eventually. This confirms once more that the heuristic from step 3 makes sense, because the more to the left side of the tree we find ourselves, the fewer leaf nodes are eliminated by step 1 and the lower the expected amount of empty visits will be.

It can be concluded that at most, all leaf nodes will be visited. Nonetheless, due to the nature of a B^+-tree based structure, this traversal will be a specific factor faster than when using no index, as the index construction guarantees spatial locality of records. If it turns out that the secondary linked list traversal only moves to the left, it was unnecessary. This property can prove valuable for improving the heuristic and how query optimizers implement it. For example, the database system could maintain the secondary linked list in a separate data structure, rather than embedding it in the leaf nodes, which would enable it to load large parts of the linked list and to skip across different nodes without actually needing to fetch those leaf nodes from the hard disk, until a jump to the right of the starting point is detected.

Finally, let us try to estimate the characteristics of a realistic data set in order to be able to determine how useful an IBPT index would generally be. The knowledge regarding a datum will always be one of the following three cases:

1. the value is precisely known,
2. the value is unknown, or
3. the value is vaguely known.

As already discussed, unknown values should be omitted from the index and precise values actually improve the performance of IBPT, because they have a positive influence on the efficiency of the secondary list. Though certainly not always true, assuming on the one hand that the amount of vague values is relatively small compared to the size of the entire data set, and on the other hand that the fuzziness is generally limited (i.e. the intervals are small), IBPT has a performance that is comparable to that of 2ABPT. Though IBPT can never perform better than 2ABPT, it will be almost as efficient under these circumstances. The largest advantage of IBPT lies in the fact that it is a single index structure, whereas 2ABPT consists of two separate CBPT indices that must be kept consistent and will take up almost twice as much space. Again, the importance of the trade off between space and efficiency can be a determining factor to choose for IBPT.

5.5 Conclusions

In this work, we have proposed Interval B^+-trees, a novel approach to indexing interval data, and have studied their applicability when it comes to performing preselection in order to evaluate fuzzy queries on uncertain data. In contrast to 2ABPT, the fastest, state of the art index for such data, IBPT requires about only half as much storage space. Though IBPT can never be strictly faster than 2ABPT, in many realistic cases, it will be comparably fast. Its main advantage is that it is a single data structure, which not only reduces its size but also simplifies maintenance operations. Compared to other B^+-tree based single index structures (such as a regular composite BPT index), IBPT is typically faster or comparable, but never slower. It has been established through experiments that the performance of IBPT is mostly sensitive to the fuzziness of the data set. More precisely, the fuzzier the data, the worse the performance. However, it has been discussed how data with a high degree of fuzziness (i.e. records about which only little information is known) can be omitted from the index and treated separately, which partially mitigates this issue.

Bibliography

[1] Alfred V. Aho and Jeffrey D. Ullman. *Data structures and algorithms*. Pearson, 1983.

[2] C. D. Barranco, Jesús R. Campaña, and J. M. Medina. "A Low Implementation Cost Alternative for Indexing Fuzzy Numerical Data". In: *2007 IEEE International Fuzzy Systems Conference*. July 2007, pp. 1–6. DOI: 10.1109/FUZZY.2007.4295627.

[3] Carlos D. Barranco, Jesús R. Campaña, and Juan M. Medina. "An Indexing Technique for Fuzzy Numerical Data". In: *Scalable Uncertainty Management*. Ed. by Henri Prade and V. S. Subrahmanian. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 187–200. ISBN: 978-3-540-75410-7.

[4] Carlos D Barranco, Jesús R Campaña, and Juan M Medina. "Indexing Fuzzy Numerical Data With a B^+ Tree For Fast Retrieval Using Necessity-Measured Flexible Conditions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 17.01 (2009), pp. 1–23.

[5] Carlos D Barranco, Jesús R Campaña, and Juan M Medina. "On the behavior of indexes for imprecise numerical data and necessity measured queries under skewed data sets". In: *International Conference on Flexible Query Answering Systems*. Springer. 2011, pp. 485–496.

[6] Carlos D Barranco, JR Campaòa, and Juan Miguel Medina. "A B^+-tree based indexing technique for fuzzy numerical data". In: *Fuzzy Sets and Systems* 159.12 (2008), pp. 1431–1449.

[7] Carlos D Barranco and Sven Helmer. "An impact ordering approach for indexing fuzzy sets". In: *Fuzzy Sets and Systems* 196 (2012), pp. 33–46.

[8] Carlos D Barranco and Sven Helmer. "Increasing the Performance of Fuzzy Retrieval Using Impact Ordering." In: *IFSA/EUSFLAT Conf.* 2009, pp. 957–962.

[9] Patrick Bosc and M Galibourg. "Indexing principles for a fuzzy data base". In: *Information Systems* 14.6 (1989), pp. 493–499.

[10] Birgit Boss and Sven Helmer. "Indexing a fuzzy database using the technique of superimposed coding-cost models and measurements". In: *Technical reports* 96 (1996).

[11] Ronald Fagin et al. “Extendible hashing—a fast access method for dynamic files”. In: *ACM Transactions on Database Systems (TODS)* 4.3 (1979), pp. 315–344.

[12] Witold Litwin. “Linear hashing: a new tool for file and table addressing.” In: *VLDB*. Vol. 80. 1980, pp. 1–3.

[13] Chengwen Liu et al. “Performance evaluation of g-tree and its application in fuzzy databases”. In: *Proceedings of the fifth international conference on Information and knowledge management*. ACM. 1996, pp. 235–242.

[14] Juan Miguel Medina, Carlos D Barranco, and Olga Pons. “Evaluation of indexing strategies for possibilistic queries based on indexing techniques available in traditional RDBMS”. In: *International Journal of Intelligent Systems* 31.12 (2016), pp. 1135–1165.

[15] Juan Miguel Medina, Carlos D Barranco, and Olga Pons. “Indexing techniques to improve the performance of necessity-based fuzzy queries using classical indexing of RDBMS”. In: *Fuzzy Sets and Systems* 351 (2018), pp. 90–107.

[16] Juan Miguel Medina et al. “Building and evaluation of indexes for possibilistic queries on a fuzzy object-relational database management system”. In: *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE. 2017, pp. 1–6.

[17] Israel Spiegler and Rafi Maayan. “Storage and retrieval considerations of binary data bases”. In: *Information processing & management* 21.3 (1985), pp. 233–254.

[18] Adnan Yazici and D Cibiceli. “An index structure for fuzzy databases”. In: *Fuzzy Systems, 1996., Proceedings of the Fifth IEEE International Conference on*. Vol. 2. IEEE. 1996, pp. 1375–1381.

[19] Adnan Yazici and Dogan Cibiceli. “An access structure for similarity-based fuzzy databases”. In: *Information Sciences* 115.1-4 (1999), pp. 137–163.

Chapter 6

Conclusions

Undisputably, computers are already extremely powerful and capable of solving problems that would be unfeasible to solve manually. Nevertheless, they are still confined to their rather limited mode of operating on precise data, information and knowledge and hence struggle when faced with problems that involve imperfect information. Fuzzy set theory provides us with a sound theoretical framework for bridging this gap between discrete, precise computers and their natural, fuzzy, human operators. Using fuzzy sets, it would be possible to store uncertain information in databases and to query this information with fuzzy query terms. However, introducing fuzzy sets has a lot of consequences. Existing techniques that rely on precise data have to be revisited. The work in this dissertation is situated in this area of research, and covers only some of the topics that have to do with the consequences of introducing imperfect information in the world of computers. This concluding chapter summarizes the novel scientific contributions that were presented in this thesis and lists some remaining research challenges that could be interesting to research in the future, structured according to the chapters in which they were discussed. Hereby, we also discuss how the three research questions presented in the beginning of this work have been addressed and answered. These research questions were:

1. How should the evaluation results of a flexible query on uncertain data be represented *truthfully*?

2. What is the impact of incorporating uncertain data and flexible querying on multi-criteria decision making (more precisely: on aggregation techniques)?

3. Today's systems are fast, and this is partly due to indexing techniques that rely on precise data. Working with uncertain data prevents the application of traditional indexing techniques. How can we adapt traditional indexing techniques so that they can be applied on uncertain data?

6.1 Flexible evaluation of uncertain data

The first research question that we wanted to answer was how to truthfully represent the results of evaluating flexible queries on uncertain data. A literary study has shown that it is not trivial to reconcile the current techniques for flexible querying on the one hand and for dealing with uncertain data on the other. Both concepts change the way a classical result set is represented from a regular set to a fuzzy set, but in a different way. The most common approach to combine both uses two degrees, possiblity and necessity, to try to express the evaluation result. This numerical approach makes it simple to compare entities, but as we have shown, also enforces a particular sorting order that can be counter-intuitive. Even in cases where the resulting order does feel intuitive, this technique is not flexible enough to allow for modification by other agents that might hold a different opinion on how the results should be ordered. This inflexibility is the result of a conscious trade-off that was made by its inventors, which sacrifices configurability for simplicity in order to be more directly compatible with existing systems.

With the introduction and study of suitability distributions, we have shown that there does not necessarily have to be a trade-off. A suitability distribution is essentially a functional relation between grades of suitability and grades of possibility. It captures that the degree to which an uncertain value satisfies a criterion, is likely also uncertain. Suitability distributions are very flexible and do not enforce a particular order. They reflect that, under uncertainty, a single, true order does not always exist. Different agents might have different opinions on what the correct order should be, based on how they personally estimate the risk of relying on a value that is uncertain. Suitability distributions can be compared, given that a specific tolerance towards uncertainty is first specified. This way, different agents may obtain different orders. In case there is no uncertainty, there is no difference compared to working with a traditional, non-fuzzy approach and the chosen tolerance has no influence on the results.

Suitability distributions are furthermore truthful. The entire point of creating a suitability distribution is to reflect all possible values of an uncertain attribute. Any loss of information is minimized by keeping uncertainty and suitability separated. This results in a two-dimensional solution that takes the form of an uncertainty distribution similar to the one that is used in the framework that models the underlying attribute value uncertainty. Because of this similarity with the underlying framework, suitability distributions are immediately well-known and come with a lot of useful properties.

For all of its benefits, suitability distributions also have downsides. Though the observed properties and applicability of suitability distributions are always applicable, we have limited our research to only the “easy” cases, where both uncertainty and user preferences are represented by simple, trapezoidal distributions. This has allowed us to simplify our calculations and made it easier to compare suitability distributions to existing techniques. The main reason we chose to limit ourselves, is because in order to tap into the full potential of suitability distributions, we would need to perform analytical calculations, which

are not trivially done on a computer. These simplifications made it possible to reduce the calculations to a few numerical computations. Clearly, it would be wise to keep studying suitability distributions in a more broad setting, allowing for more general distributions.

Aside from that, there are still many other things left to research in the field of uncertainty in flexible querying. There seems to be an ever-growing list of frameworks to express uncertainty in, and with it, more and more possible fields of application that could serve as a novel way to expand and test the theory behind suitability distributions. Aside from technical complexities, most questions that come to mind deal with integration in existing systems. Using suitability distributions would require changing the way we ask questions. It would require that a whole array of challenges that have already been overcome, be revisited and re-evaluated. We have already covered some examples, like sorting and comparing results, but there are others. By replacing graded results by uncertainty distributions, the entire calculus regarding multi-valued logic is uprooted. Entire fields like aggregation, visualization and indexing are impacted and must be investigated. One example of an area of application is flexible controllers (rule-based systems), wherein computers are employed to automate certain behavior (e.g. self driving cars). These systems often rely on many sensors that may at some point produce inaccurate readings or even fail alltogether. Another possible application lies in imputation. This technique concerns itself with trying to replace missing values by realistic estimates (in cases where the absence of the value does not indicate inapplicability but rather incomplete knowledge). A common critique on imputation is that, no matter which particular implementation is used, introducing certain value impacts analyses that are carried out on the data. Aside from making it possible to perform analyses in the first place, different imputation techniques influence different analyses. Instead of replacing missing values by particular estimates, replacing them by uncertain values (possibly having a particular distribution) is more truthful in the sense that it reflects that the value was imputed and should be treated as uncertain. Suitability distributions might then find applicability in the analyses performed on datasets that were imputed in such a way.

6.2 Dealing with uncertainty in aggregation

Realistic decisions are rarely taken based on the value of a single attribute. Typically, many different aspects are taken into account when evaluating a set of alternatives. This complicates the process of finding the best solution, because comparing alternatives is no longer straightforward. Of the different techniques that serve to support multi-criteria decision making, aggregation is the most popular. In aggregation, the evaluation of an alternative is broken down into first evaluating the individual per-attribute criteria and then aggregating those outcomes into a single, global rating. Despite being an inherently lossy practice (as multiple indicators of information are intentionally aggregated into a single indicator), it is popular due to the fact that it feels intuitive and is quite flexi-

ble. Indeed, aggregation techniques are capable of sorting alternatives in ways that are not possible when using other, lossless approaches (e.g. lexicographical ordering). This makes aggregation an active research area of its own, in which several different approaches for performing flexible, semantically rich aggregation exist. These approaches compete to try to be as flexible as possible, and even though they all have the same goal, they are not true alternatives: some techniques have features that others do not.

In that regard, we have proposed a way to extend fuzzy integration techniques in such a way that they can realize the partial absorption, an operator that is otherwise only found in the generalized conjunction/disjunction. The way we have done this is by allowing the weights that define the behavior of the fuzzy integral to be functions rather than scalars. We have shown that by choosing functions that satisfy particular constraints, the conjunctive partial absorption can be implemented. In the future, this research could be extended by studying other advanced aggregators such as the disjunctive partial absorption, compound absorption aggregators and more.

Another reason due to which aggregation is further complicated, is by considering that data might be uncertain. Given our contributions to the modelling of uncertainty in flexible evaluation of uncertain data, we have also studied the effects of introducing suitability distributions in aggregation frameworks. It was discussed that suitability distributions can be aggregated directly, albeit generally by having to resort to numeric approximation techniques. However, we have also shown that there is merit to defuzzifying the distributions before aggregating them, as doing so makes it possible to specify different tolerance levels towards the uncertainty of individual criteria. In addition, the fact that suitability distributions inherit the properties of the uncertainty framework in which the data are expressed, results in different families of suitability distributions that may not be directly comparable. For example, the uncertain suitability of a stochastic variable (expressed using probability theory) can not directly be aggregated with the uncertain suitability of an ill-known variable (expressed using possibility theory). It has to be investigated further whether or not these can be compared and if so, how this is done properly. Though there has been research connecting both frameworks, it has not yet been studied in the context of suitability distributions. Moreover, there are other frameworks, too, for which a way to compare and translate their way of representing uncertainty has yet to be investigated. This is clearly a very rich area when it comes to research opportunities.

Though it may seem like defuzzifying first solves some of these problems, it creates a different problem: how to aggregate indicators of uncertainty? There seems to be no standardized way to do so, but one common suggestion is to aggregate the grades of suitability and those of uncertainty separately from each other. We have studied this idea in the context of the generalized conjunction/disjunction and have found that it is possible to aggregate uncertainty indicators, though not entirely independently of the suitability aggregation. First, the suitability grades should be aggregated individually, as if there was no uncertainty, and then the uncertainty grades can be aggregated based on weights

that are derived from the "dominance" of their corresponding suitability grades (i.e. how big their influence is on the suitability aggregation). Furthermore, we have discussed that the uncertainty indicators used in the aggregation should all share similar semantics for the results of the aggregation to make sense. Our research in this area has been limited to a study of binary aggregators (i.e. of two inputs), leaving a lot of unexplored potential for future research.

6.3 Indexing of uncertain attribute values

Indexing is a common practice to speed up query evaluation when looking up information in datasets. Common indexing techniques rely on a natural (or otherwise specified) order among the values of an attribute's domain, but such a (weak) total order no longer necessarily exists when introducing uncertain attribute values.

When building a result set from a dataset that contains uncertain values, one can rely on preselection, which is a principle that uses two tests in order to determine if an attribute value could possibly overlap with a specified fuzzy preference. In case that an attribute value satisfies either test, it can be rejected safely. This preselection procedure is the key to indexing techniques for uncertain data, and can be employed to avoid having to scan the entire dataset and having to evaluate the two tests for every entity.

Our novel contribution in this area is the interval B^+-tree: a new indexing technique based on B^+-trees, which are themselves a popular indexing method for regular data. B^+-trees use a linked list to traverse the indexed attribute values quickly. Interval B^+-trees introduce a second linked list to traverse the indexed attribute values according to an alternative order. Combined with the original linked list, these two linked lists provide an efficient way to apply the tests on all indexed values, such that large parts of the dataset can be rejected very quickly. Though there exist other techniques that are typically faster than interval B^+-trees, they are always larger in size and more difficult to maintain. However, in many conditions, interval B^+-trees will be able to perform comparably well. Interval B^+-trees provide an interesting extra tool for database experts that allow them to optimize aggressively, leveraging storage space for execution speed.

There are still a few loose ends that can be investigated further. For example, the proposed method relies on a heuristic which has an impact on the efficiency of the index. In our research, we have only investigated a simple heuristic, and we are confident that better heuristics must exist. Another example is that the secondary linked list could be optimized further. In its current form, it is required that it is traversed until the end, though it would be possible to stop early given that a certain condition is satisfied. This would require that additional information is stored in the index, but it might prove to be beneficial to do so.

6.4 Final remarks

It should be clear by now that there is still a long way to go before computers can be considered as flexible as humans when it comes to handling imperfect information. Perhaps, they may never reach this point, due to the fundamental differences in how they treat data, information and knowledge. However, with the contributions that were presented in this dissertation, it is shown that there might be a partial solution, that allows decision makers to intuitively model their evaluation logic and apply it on uncertain data, without requiring that computers modify the way they store data. There are still plenty of loose ends to pick up, though, as many questions remain unanswered.

6.5 List of publications

1. Robin De Mol, Ana Tapia Rosero, and Guy De Tré. "An approach for uncertainty aggregation using generalised conjunction/disjunction aggregators". In: *16th World Congress of the International Fuzzy Systems Association (IFSA); 9th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT)*. vol. 89. Atlantis Press. 2015, pp. 1499–1506

2. Robin De Mol, Antoon Bronselaer, and Guy De Tré. "Evaluating flexible criteria on uncertain data". In: *Fuzzy Sets and Systems* (2017)

3. Robin De Mol and Guy De Tré. "Representing Uncertainty Regarding Satisfaction Degrees Using Possibility Distributions". In: *Advances in Fuzzy Logic and Technology 2017*. Springer, 2017, pp. 597–604

4. Antoon Bronselaer, Robin De Mol, and Guy De Tré. "A measure-theoretic foundation for data quality". eng. In: *IEEE Transactions on Fuzzy Systems* 26.2 (2018), pp. 627–639. ISSN: 1063-6706

5. Christophe Billiet, Robin De Mol, and Guy De Tré. "A novel fuzzy result ranking technique". eng. In: *2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS)*. Otsu City, Japan: IEEE, 2017. ISBN: 9781509049172

6. Antoon Bronselaer et al. "Ordinal assessment of data consistency based on regular expressions". eng. In: *Information Processing and Management of Uncertainty in Knowledge-based Systems, IPMU 2016, PT II*. ed. by Joao Paulo Carvalho et al. Vol. 611. 2. Eindhoven, The Netherlands: Springer, 2016, pp. 317–328. ISBN: 978-3-319-40580-3

7. Robin De Mol, Antoon Bronselaer, and Guy De Tré. "Partial absorption aggregators with fuzzy integrals". eng. In: *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. Vancouver, Canada, 2016, pp. 424–430. ISBN: 978-1-5090-0625-0

8. Robin De Mol et al. “Data driven Xpath generation”. eng. In: *Advances in Intelligent Systems and Computing*. Ed. by P Angelov et al. Vol. 322. Warsaw, Poland: Springer, 2014, pp. 569–580. ISBN: 9783319113128

9. Robin De Mol and Guy De Tré. “Applying suitability distributions in a geological context”. eng. In: *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations*. Cadiz, Spain: Springer International Publishing, 2018. ISBN: 9783319914725

10. Guy De Tré, Robin De Mol, and Antoon Bronselaer. “Indexing possibilistic numerical data : the interval B^+-tree approach”. eng. In: *Information Processing and Management of Uncertainty in Knowledge-based Systems, IPMU 2016, PT II*. ed. by Joao Paulo Carvalho et al. Vol. 611. 2. Eindhoven, The Netherlands: Springer, 2016, pp. 305–316. ISBN: 978-3-319-40580-3

11. Guy De Tré, Robin De Mol, and Antoon Bronselaer. “On the need for explicit confidence assessments of flexible query answers”. eng. In: *Flexible Query Answering Systems, FQAS 2017*. Ed. by Henning Christiansen et al. Vol. 10333. London, UK: Springer, 2017, pp. 51–58. ISBN: 978-3-319-59692-1

12. Guy De Tré, Robin De Mol, and Antoon Bronselaer. “Handling veracity in multi-criteria decision-making : a multi-dimensional approach”. eng. In: *INFORMATION SCIENCES* 460 (2018), pp. 541–554. ISSN: 0020-0255

13. Guy De Tré et al. “Data quality assessment in volunteered geographic decision support”. eng. In: *Mobile information systems leveraging volunteered geographic information for earth observation*. Ed. by Gloria Bordogna and Paola Carrara. Vol. 4. Earth Systems Data and Models. Springer International Publishing, 2017. ISBN: 9783319708775

14. Guy De Tré et al. “Human centric recognition of 3D ear models”. eng. In: *International Journal of Computational Intelligence Systems* 9.2 (2016), pp. 296–310. ISSN: 1875-6891

15. Joachim Nielandt et al. “Wrapper induction by XPath alignment”. eng. In: *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*. Ed. by Ana Fred and Joaquim Filipe. Vol. 6. Rome, Italy: Science and Technology Publications, 2014, 107:492–107:500. ISBN: 9789897580482

16. Ana Tapia Rosero et al. “Fusion of preferences from different perspectives in a decision-making context”. eng. In: *Information Fusion* 29 (2016). Ed. by Francisco Herrera and Luis Martínez, pp. 120–131. ISSN: 1566-2535

17. Ana Tapia Rosero, Robin De Mol, and Guy De Tré. “Handling uncertainty degrees in the evaluation of relevant opinions within a large group”. eng.

In: *Computational Intelligence, IJCCI 2014*. Ed. by Juan Julian Merelo et al. Vol. 620. Rome, Italy: Springer International Publishing, 2016, pp. 283–299. ISBN: 978-3-319-26391-5

18. Vera Van Lancker et al. "Building a 4D voxel-based decision support system for a sustainable management of marine geological resources". eng. In: *Oceanographic and marine cross-domain data management for sustainable development*. Ed. by Paolo Diviacco, Adam Leadbetter, and Helen Glaves. Advances in Environmental Engineering and Green Technologies (AEEGT) Book Series. IGI Global, 2017, pp. 224–252. ISBN: 9781522507017