Giovanni Misitano

INFRINGER: a novel interactive multi-objective optimization method able to learn a decision maker's preferences utilizing machine learning

Master's thesis in Information Technology

July 3, 2020

University of Jyväskylä

Faculty of Information Technology

Author: Giovanni Misitano

Contact information: giovanni.a.misitano@jyu.fi

Supervisors: Kaisa Miettinen, and Jussi Hakanen

Title: INFRINGER: a novel interactive multi-objective optimization method able to learn a decision maker's preferences utilizing machine learning

Työn nimi: INFRINGER: uusi interaktiivinen monitavoiteoptimoinnin ongelmien ratkomiseen kehitetty menetelmä, joka kykenee oppimaan päätöksentekijän preferenssit

Project: Master's thesis

Study line: Computational sciences

Page count: 110+3

Abstract: An interactive method – INFRINGER – for solving multi-objective optimization problems is developed in this thesis. The method is able to learn a decision maker's preferences using a value function model. The value function is modelled using machine learning in conjunction with belief-rule based systems. A case study, consisting of a problem in Finnish forestation, is then conducted where a human decision maker is aided in the decision making process using the developed method. Based on the results of the case study, the developed method is assessed in its ability to aid the decision maker to reach a satisfying solution, and its ability to elicit the decision maker's preferences. Lastly, the method is briefly compared qualitatively to other similar methods in existing literature, and the viability of the method as a potential explainable model is briefly discussed.

Keywords: multi-objective optimization, decision making, preference learning, belief-rule based systems, machine learning, data-based decision support

Suomenkielinen tiivistelmä: Tässä tutkielmassa kehitetään interaktiivinen menetelmä – nimeltään INFRINGER – monitavoiteoptimoinnin ongelmien ratkaisemisen tueksi. Menetelmä kykenee oppimaan päätöksentekijän mieltymykset (preferenssit), ja esittää mieltymyksiä käyttäen arvofunktiota mallinaan. Arvofunktio mallinnetaan käyttäen koneoppia, jossa sovel-

letaan todennäköisyyksiä hyödyntäviä sääntöpohjaisia järjestelmiä. Kehitettyä menetelmää hyödynnetään tapaustutkimuksessa, jossa päätöksentekijää tuetaan Suomen metsätalouteen liittyvän monitavoitteisen optimointiongelman ratkaisemisessa. Tapaustutkimuksen tulosten pohjalta kehitetyn menetelmän kykyä tukea päätöksentekijää, ja oppia päätöksentekijän mieltymykset, arvioidaan. Lopuksi kehitettyä menetelmää verrataan lyhyesti vastaaviin kirjallisuudessa esiintyviin menetelmiin, ja menetelmän kelpoisuutta selitettävänä koneopin mallina pohditaan.

Avainsanat: monitavoiteoptimointi, päätöksenteko, preferenssin oppiminen, todennäköosyyksiin perustuvat sääntöpohjat, koneoppi, datapohjainen päätöksenteon tuki

List of Figures

List of Tables

Contents

1 Introduction

This chapter starts with a discussion of the general motivation behind this thesis in Section 1.1. After which in Section 1.2, a short introduction to machine learning is given. In Section 1.3, a brief overview of existing works discussing methods similar to the novel method developed in this is given. To conclude this chapter, the central research questions to this thesis are presented in Section 1.4, followed by the description of the general structure and outline of this thesis given in Section 1.5.

1.1 Motivation

Real life problems can often be modeled as multi-objective optimization problems with multiple conflicting criteria. Solutions to such problems are seldom unambiguous and the need of a decision maker – a real human being with an adequate expertise in a field relevant to the problem – is needed.[1] Very often, if not always, the solutions of interest consist of Pareto optimal solutions; solutions which are defined as a set where no solution can be chosen over another solution without making trade-offs in regards to one or more criteria [1]. Out of the many methods developed to aid a decision maker in finding the most satisfying option among these Pareto optimal solutions, interactive multi-objective optimization methods are of particular interest in this thesis. The focus is on interactive methods because these methods actively involve the decision maker in the solution process, which arguably improves the chances of the final solution found to match the decision maker's preferences. Examples of interactive multi-objective optimization methods can be found in [2, 3, 4, 5].

Evidently the decision maker's role is of crucial importance. The key ingredient the decision maker can provide is their preferences. Preference modelling is a research field of its own [6], which studies how to model preferences, and also the various factors affecting how preferences are constructed by humans – such as psychological and socioeconomic, but also moral

[1] In some problems, more than one decision maker are present. However, in this thesis we will not discuss problems with multiple decision makers and therefore limit the discussion to problems involving only a single decision maker.

and ethical as well. Having some methodology to build a working model of the preferences of a decision maker would allow the analysis of the process of a decision maker reaching a satisfying solution – a solution that matches the preferences of the decision maker. This would allow for a post decision making analysis of the possible factors affecting the decision maker in making the decisions, and permit for an overall assessment of the quality of the decision made, as is done in [7], for example.

As multi-objective optimization problems emerge, for example in economical [8], medical [9], technological [10], and environmental contexts [11], it is of utter importance – in the author's opinion – that if the decision maker's preferences are to be modelled, they should be modelled using transparent and explainable methodologies. Instead, if oblique methodologies are used in aiding decision makers, the margin for erroneous behavior could be too large to guarantee that the decisions made are reflective of the true underlying preferences. Malicious models, possibly stemming from the use of the oblique methodologies, could cause significant repercussion affecting society on a varying degree.

This thesis is intended to take a step, albeit small, towards the *use* of explainable machine learning models to model the preferences of a decision maker. An emphasis is on the *use* of such models, because in this thesis the behavior of the developed model is not attempted to be explained. The explainable nature emerges as a feature of the underlying machine learning model used in the method developed in this thesis.[2]

[2]Recently, the feat discussed in this paragraph was successfully accomplished. In [12], a machine learning model inherently explainable – an *interpretable* model – was used in place of a proprietary and oblique model used for decision support in recidivism predictions. On top of being explainable, the interpretable model performed just as well as the proprietary model. However, because of its explainable nature the interpretable model was argued by the author to be objectively better than its proprietary counterpart. The reason for the presented argument was partly the fact that the explainable nature of the interpretable model allows decision makers to not just better understand why a certain prediction is given by the model, but to also incorporate his/her own expertise with the prediction to ultimately produce better decisions.

1.2 Why machine learning?

Machine learning is at the heart of the method developed in this thesis. However, the focus of this thesis should not stray away from the main motivation: how to best aid a decision maker to reach a satisfying solution in a multi-objective optimization problem? Learning the preferences of the decision maker is but just one approach to answer this problem. To keep the discussion in this thesis focused on the main motivation, the discussion of the theory of machine learning is kept to a bare minimum. Therefore, a few central concepts are presented in this section accompanied with a brief discussion on relevant topics to give the reader the necessary tools to follow along. For a detailed introduction to machine learning, see [13] for example.

The goal of machine learning is to enable a computer to perform some task without explicitly telling the computer how to perform it. Instead, a machine learning model is often used, which has been trained to perform a specific task. Training in the context of machine learning can mean many things, but what is generally meant by training, can be understood as varying a mathematical model's internal parameters to match a desired behavior. When a model is trained several times with its behavior closing to what is desired, the model is said to be learning.

To clear some of the abstractions in the previous paragraph, consider the case of simple linear regression: fitting of a straight line

$$y(x) = ax + b, \tag{1.1}$$

where $a, b \in \mathbb{R}$ are the slope and y-intercept parameters respectively, and x, y are the independent and dependent variables, respectively. In fitting the line (1.1) to an observed set of presumably linearly correlating data, a simple linear model is trained by finding the adequate slope and intercept values. This produces a linear function able to reproduce the behavior seen in the data, and can be used to make estimations in regions where no explicit data has been observed before. This is evident for example in predicting stock prices. In stock price prediction, prices are observed at different times, and based on these observations, a linear

model is constructed. The linear model is then able to predict the stock prices at a future, unobserved time. The more data is observed, the more accurate the trained model will be.

After fitting the line (1.1) to the observed data, the linear model is considered to be trained. If the linear model has been previously fitted to a sparse set of data, and then fitted again to a more dense set of data producing better results, the model is said to be learning. This idea can be generalized to a plethora of different mathematical models, not just linear regression. An example of such models are belief-rule based systems [14] discussed in Chapter 3, which are used in the novel interactive multi-objective optimization method developed this thesis.

Before concluding this short introduction to machine learning, it is worth discussing the explainability of machine learning models. After fitting the line (1.1), the meaning of the trainable variables, the slope and the intercept, are known: the greater the slope, the faster the dependent variable grows as the independent variable is increased.[3] The intercept value is the value of the dependent variable when the dependent variable is zero. This means that the behavior of linear model in (1.1) can be explained. For example, if the slope is very high, it is explainable why the dependent variable y increases a lot when the independent variable x is increased only slightly.

However, more complicated models like deep neural networks [15] have sometimes thousands of trainable variables with a mostly unknown and unexplainable meaning. This kind of complex models are called black-box models. They are named so because the models often manifest themselves as black-boxes, which take some input and then map it to some output. It is not explicitly known how the model maps the input to an output. Black-box models are often said to be oblique machine learning models in contrast to transparent models, whose inner workings are understood to a degree which allows for an explanation of how the model works to be given – either by the model itself or some expert with the adequate knowledge. Indeed, these transparent models are also called explainable models.

The main motivation of using belief-rule based systems as the machine learning model in the developed method in this thesis, is the potentially explainable nature of belief-rule based sys-

[3] Or decreased, if the relationship of the independent variable is inversely proportional to the dependent variable.

tems. Recent discussion on the explainable nature of belief-rule based systems can be found in [16] and [17]. Another approach is to apply a meta-model on top of an oblique model in an effort to try and produce explanations on the model's behavior. A recent example of this methodology is the Local Interpretable Model-Agnostic Explanations (LIME) discussed in [18]. The advantage of LIME is that it can be used with powerful black-box models, which have already been trained, to produce explanations on their behavior. That being said, based on the argument given in a recent article [12], explaining black-box models might not lead to explanations reflective of the true reasons behind a prediction made by a black-box models. Instead, *interpretable* machine learning models, which are inherently also explainable should be used. In this thesis, interpretable machine learning models will not be discussed further.

Because of the nature of decision making involving real human beings, available data on the decision making process if often sparse and powerful models like deep neural networks are not usable because they require large amounts of data to be trained; deep neural networks require usually tens, if not hundreds of thousands of samples to be trained. Yet, belief-rule based systems do not necessarily require large amounts of data. They can work very well with just hundreds of samples, or even less. Moreover, belief-rule based systems are built on the basis of principles and rules, which are not necessarily dependant on raw data and can therefore be also inferred using available qualitative knowledge.

Therefore, machine learning is used in this thesis in an effort to try and learn the preferences of a decision maker because machine learning allows to build computational models, which once trained can perform a task without special supervision. In this thesis, belief-rule based systems are chosen as a machine learning model because they offer a promising potential for explainability. Explainability is hold in high regard by the author of this thesis because of the moral implications of using machines to assist real humans in real life decision making scenarios.

1.3 Examples of decision maker's preference learning in existing literature

Some existing works similar to the novel method developed in this thesis, which model the preferences of a decision maker utilizing machine learning during an interactive multi-objective optimization process are discussed in this section. Most of the presented works involve multi-objective optimization methods, which make use of evolutionary algorithms. These methods are known as (interactive) evolutionary multi-objective optimization methods. The works discussed in this section have been chosen because they contain methods similar to the method developed in this thesis. This custom has led to an over-representation of methods involving evolutionary multiobjective optimization.

In [19], the brain-computer evolutionary multi-objective optimization (BC-EMO) algorithm is presented. BC-EMO is claimed to be able to learn an arbitrary value function from preference information gained interactively from a decision maker. It uses simple pair-wise comparisons when eliciting preference information from the decision maker, and it uses the learned value function to guide the search process towards more preferred solutions. BC-EMO is used to elicit preference information from which new solutions are generated. As its underlying machine learning model, BC-EMO uses support vector machines.

In [7], the preference learning based evolutionary multi-objective optimization (PLEMOA) method is proposed. PLEMOA makes use of pair-wise comparisons to gather information to learn an ideal solution corresponding the preferences of a decision maker. The method aims to guide the optimization process to produce solutions that are interesting to the decision maker by generating solutions close to the learned ideal solution. The learned preference model can also be used to predict the decision maker's future choices. Maximum likelihood estimation is used to learn the ideal solution of the decision maker. The performance of PLEMOA is compared to PL-NSGA2 proposed in [20], which is similar to PLEMOA, but limits the pair-wise comparisons to be done between solutions on the Pareto optimal solution set instead of using an intermediate solution to guide the optimization process, like is done in PLEMOA.

A framework consisting of three different variants of preference enhanced evolutionary

multi-objective optimizer (NEMO) is presented in [21]. The three variants are NEMO-0, NEMO-I, and NEMO-II. Common to all variants, ordinal regression is used to infer one or more additive value functions based on a given ranking of alternatives in a finite set of objective vectors, for example. The required ranking information is elicited from a decision maker using pair-wise comparisons. The NEMO variants mutually differ in the kind of value function used to rank solutions in a population of solutions.

In [22] a multiplicative value function is proposed, which aims to combine the preferences from multiple decision makers for learning appropriate hyper parameters for a machine learning model during its production. The proposed value function aims to incorporate the multiple conflicting inputs from different decision makers. To elicit preferences, pair-wise comparisons are used. The underlying tool to fine tune the free parameters of the proposed value function is sequential model-based optimization (SMBO).

The method proposed in [23] is based on radial basis functions and inverse distance weighting for exploring the space of decision variables, where pair-wise comparisons are used to elicit a decision maker's preferences. The method is also applied to a multi-objective optimization problem and it makes no assumption of the form of the objective functions. The method interactively learns a surrogate model to reflect the decision maker's preferences.

1.4 Research questions

The purpose of this thesis is to explore and answer the following research questions:

1. Can belief-rule based systems be used to learn and model a decision maker's preferences for a multi-objective optimization problem?
2. Can the learned model for the preferences predict solutions that are in accordance with the decision maker's preferences?
3. Is it possible to combine the ideas explored in questions 1 and 2 into an interactive multi-objective optimization method to aid a decision maker make decisions in a real-life decision making problem with multiple conflicting criteria?
4. Is the resulting preference model explainable?

1.5 Structure of this thesis

In Chapters 2 and 3 a basic introduction to the central concepts in multi-objective optimization and to belief-rule based systems is given, respectively. Based on these concepts, Chapter 4 presents the INFRINGER method: a novel interactive method for solving multi-objective optimization problems, which is able to learn a decision maker's preferences. A case study involving a problem in Finnish forestation is then conducted in Chapter 5. The results of the case study are discussed in Chapter 6, where the viability of the INFRINGER method for aiding a human decision maker in a decision making process is assessed and the method is also briefly compared to other similar methods in existing literature in a qualitative fashion. Concluding this thesis, in Chapter 7 the research questions presented in Section 1.4 are answered and a few potential research topics for future research are suggested.

2 Multi-objective optimization

In this chapter, a general introduction to the central concepts in multi-objective optimization is given. An informal introduction is given in Section 2.1 first, followed by a more formal discussion of the central concepts in multi-objective optimization in Section 2.2. Methods for solving multi-objective optimization problems are discussed in Section 2.3, and in Section 2.4 it is defined what is meant by a decision maker and how his/her preferences can be modelled. This chapter is concluded with Section 2.5 where the concept of an interactive method is defined, and the questions of why interactive methods are important and how they differ is discussed. For a more detailed discussion on multi-objective optimization, see [1] and [24].

2.1 What is multi-objective optimization?

Consider the following cases:

1. A CEO of a multi-billion corporation has to decide what course to take in terms of managing her business during the next financial year. She wishes to maximize the profit of her corporation, and to keep the stake holders happy. She cannot make too many cuts in the yearly budget without considerate repercussions.
2. A student is at his wit's end: should he take a quickie loan[1] to afford proper food, or should he swallow his pride and fare with cup noodles for the rest of the month?
3. A recruiter for a local supermarket has a couple of possible candidates available for an open cashier job position. Who should the recruiter choose? A more experienced candidate will require a higher pay, but a less experienced, and cheaper to employ, candidate may not be efficient enough in performing the job.

The CEO, student, and recruiter are all very different people in very different situations, but what connects them is that they are all trying to make decision based on multiple conflicting criteria.

[1] A small loan that is quick and easy to get, but has a very high interest rate.

In the previously considered cases, the CEO, student, and recruiter are all in the position of a decision maker: they have to make a decision based on different choices available, each with its trade-offs. The student, for example, has to decide whether to take a small loan now to afford proper food, but with interests to be paid in a near future; or not to take a loan resulting in saved money, but having to cope with miserable food. It can be said that the student has two criteria, or objectives, he is interested in: Minimizing his financial losses, and maximizing the quality of food. The student has also realized to his knowledge or not, that he cannot have gourmet food without paying a high cost, and he cannot pay pennies and expect a filet mignon. There is a trade-off between food quality and the money in his bank account; gaining in quality of food will lead to a loss of capital, and trying to save money will result in deterioration of food quality. The CEO and recruiter are facing similar dilemmas, but with different criteria and trade-offs.

This kind of decision making based on multiple conflicting criteria is at the heart of multi-objective optimization. This thesis has a special focus on modelling the decision maker's preferences. The preferences guide the decision maker, and understanding these preferences can help in analyzing why the decision maker has made a particular decision.

2.2 Optimization

2.2.1 Single-objective optimization

Before a multi-objective optimization problem can be formally defined, the general case of a single-objective optimization problem is presented in this section. The concepts presented for single-objective optimization can then be generalized to multi-objective optimization.

In a single-objective optimization problem, an objective is usually defined as a scalar valued function of one or multiple independent variables. Both the function and variables are assumed to be real valued in this chapter, but they could be also very well be integer valued or be part of discrete sets. However, because the real-valued case can be trivially generalized to the integer and discrete cases, the discussion here is limited to the real-valued case only.

Formally, a single objective function can be defined as

$$f : \mathbb{R}^n \to \mathbb{R}, \tag{2.1}$$

where n denotes the number of dimensions in the domain of the objective function f, or in other words, the number of independent decision variables.

The decision variables for an objective function defined in (2.1) can be defined as a vector, called a decision variable vector

$$\mathbf{x} = \{ x_i \mid \forall i \in [1, n] \}, \tag{2.2}$$

where x_i denotes the value of the ith decision variable. In a single-objective optimization problem, the variables of an objective function are often subject to constraints defined as part of the problem. These constraints can be inequality constraints or equality constraints defined as

$$g : \mathbb{R}^n \to \mathbb{R} : \; g(\mathbf{x}) \leq 0, \tag{2.3}$$

and

$$h : \mathbb{R}^n \to \mathbb{R} : \; h(\mathbf{x}) = 0, \tag{2.4}$$

respectively, and the problem formulation may contain one or more of each of these constraints. The constraints in a single-objective optimization problem are used to define a feasible space for the decision variable vectors $\mathbf{x}$, which is denoted as the feasible set X of decision variable vectors. Therefore, the feasible decision variable vectors of a single-objective optimization problem can be defined as

$$\mathbf{x} \in X \subseteq \mathbb{R}^n. \tag{2.5}$$

In the case of $X = \mathbb{R}^n$ in (2.5), the underlying optimization problem is said to be unconstrained, while in the case of $X \subset \mathbb{R}^n$, the problem is said to be constrained.

Based on the concepts defined so far, the full definition of a single-objective optimization problem can be given as

$$\min_{\mathbf{x} \in X} f(\mathbf{x}), \tag{2.6}$$

where the objective function f is to be minimized, and the decision variable vector $\mathbf{x}$ must be part of the feasible set X. The choice of whether an objective function should be maximized or minimized stems from the nature of the objective in the optimization problem: profits are to be maximized, and losses are to be minimized, for example. If need be, a minimization problem can be transformed into a maximization problem by multiplying the objective function by -1. The minimization problem (2.6) is therefore equivalent to

$$\max_{\mathbf{x} \in X} -f(\mathbf{x}). \tag{2.7}$$

The solution to a single-objective optimization problem is then defined as some decision variable vector $\mathbf{x}^* \in X$, which satisfies either (2.6) or (2.7). The objective functions is said to result in its optimal value when evaluated with $\mathbf{x}^*$ as its argument.

In some cases there might be multiple solutions to a single-objective optimization problem, for example in the case of minimizing the objective function $f(x) = |x^2 - 1|$ both solutions $x_1^* = 1$ and $x_2^* = -1$ minimize the function. In this case, the solutions x_1^* and x_2^* are considered to be equally preferable, meaning there is no clear benefit[2] in choosing one solution over the other.

[2]If there was a clear benefit in choosing one solution over the other, it should be formulated in either the objective function of the problem or as a constraint of the problem.

2.2.2 Multi-objective optimization

In multi-objective optimization the number of objectives to be optimized is greater than one. These objectives are to be optimized, either minimized or maximized simultaneously. The objective functions of a multi-objective optimization problem can be either understood as a vector valued function, or as a set of multiple scalar valued functions. Formally, the objective function of a multi-objective optimization problem can be defined as

$$\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m : \ \mathbf{f}(\mathbf{x}) = \{ f_i(\mathbf{x}) \mid i \in [1, m] \}, \tag{2.8}$$

where m denotes the number of objectives or the dimension of the objective space in a multi-objective optimization problem, and f_i denotes the individual objective function components in $\mathbf{f}$, with each f_i being a scalar valued function. It is important to note that the decision variable vector $\mathbf{x}$ in (2.8) is the same for all individual objective functions f_i. In this thesis, the objectives of a multi-objective optimization problem is always understood as a set of multiple scalar valued functions, each defined like (2.1).

The constraints that define a feasible set (2.5) in a multi-objective optimization problem are defined equivalently to (2.3) and (2.4). In other words, a multi-objective optimization problem is analogous to a single-objective optimization problem defined in (2.6) and (2.7), but differs in the number of objectives. Therefore, a multi-objective optimization problem can be defined as

$$\min_{\mathbf{x} \in X} \mathbf{f}(\mathbf{x}) = \{ f_i(\mathbf{x}) \mid i \in [1, m] \}. \tag{2.9}$$

It is typical that all the objective function in a multi-objective optimization problem are defined to be either minimized or maximized. Any single objective function can always be transformed into its equivalent counterpart, which is evident from (2.6) and (2.7).

2.2.3 Central concepts in multi-objective optimization

The problem at the core of defining what can be considered to be an optimal solution to a multi-objective optimization problem defined in (2.9), is that the objective functions are seldom independent; the objective functions in a multi-objective optimization problem are usually in conflict.[3] This means that when one of the objective functions attains its optimal value for some optimal solution $\mathbf{x}_1^* \in X$, it is very likely that some other objective function defined in the same problem does not attain its optimal value. Then, choosing a new optimal solution $\mathbf{x}_2^* \in X$ might result in some other objective function attaining its optimal value and for the first objective function to deteriorate in its value.

In a multi-objective optimization problem, a solution is not unambiguous, but rather ambiguous, and the concept of a single solution being optimal does not make sense. Therefore, the concept of a Pareto optimal solution is defined next to address this issue.

The image of the feasible set is defined as

$$Z = \mathbf{f}(X), \tag{2.10}$$

with each of its members being defined as an objective vector $\mathbf{z} = \mathbf{f}(\mathbf{x}),\ \mathbf{x} \in X$.

Consider a minimization problem with m objective functions, and a feasible decision variable vector $\mathbf{x} \in X$ with the image $\mathbf{z} = \{\, z_i \mid i \in [1,m] \,\} \in Z$. Stating that the decision variable vector $\mathbf{x}$ is a Pareto optimal solution, is equivalent to the following statement:

$$\begin{aligned} & \mathbf{x} \text{ is a Pareto optimal solution} \\ \text{iff} \quad & z_i \leq \tilde{z}_j \ \forall\, i,j \in [1,m] \quad \text{and} \quad z_i < \tilde{z}_j \ \exists\, i,j \in [1,m]; \\ \text{for all} \quad & \tilde{\mathbf{z}} = \mathbf{f}(\tilde{\mathbf{x}}) = \left\{\, \tilde{z}_j \mid j \in [1,m] \,\right\}, \\ \text{where} \quad & \tilde{\mathbf{x}} \in X \setminus \mathbf{x}. \end{aligned} \tag{2.11}$$

In other words, (2.11) implies that no other objective vector $\tilde{\mathbf{z}}$ may exist in Z, which has

[3] If the objective functions are not in conflict, then they can be optimized independently, and the optimal solution is optimal in regard to every objective function.

strictly better objective function values for all objectives if $\mathbf{x}$ is a Pareto optimal solution. Therefore, $\tilde{\mathbf{z}}$ must be worse at least in one objective function value compared to $\mathbf{z}$. The condition for Pareto optimality in (2.11) can be extended to a maximization problem by simply negating z_i and $\tilde{z}_j$ in the statement.

The concept of a Pareto optimal solution can be also extended to the concept of a Pareto optimal solution set. The Pareto optimal solution set consists of all solutions $\mathbf{x} \in X$, which satisfy condition (2.11). In this thesis the image of the Pareto optimal solution set is simply referred to as the Pareto optimal set. It follows that the Pareto optimal set Z^{Pareto} is a proper subset of the image of the feasible region: $Z^{\text{Pareto}} \subseteq Z$. The member of the Pareto optimal set $\mathbf{z} \in Z^{\text{Pareto}}$ are referred to as Pareto optimal objective vectors.

It is also useful to define the concept of dominance. An objective vector $\mathbf{z}_1$ is said to dominate another objective vector $\mathbf{z}_2$, both being part of some subset of Z, if the solution $\mathbf{z}_1$ is strictly better in at least one objective value and better or equal in the other values. This is similar to condition (2.11) if in the condition $\mathbf{z}$ is replaced by $\mathbf{z}_1$ and $\tilde{\mathbf{z}}$ by $\mathbf{z}_2$. If an objective vector is not dominated by any other objective vector in the same subset of Z, then that objective vector is said to be non-dominated in that subset. It follows from the definition given for Pareto optimality in (2.11) that the Pareto optimal set Z^{Pareto} is a set of mutually non-dominated objective vectors – meaning that none of the objective vectors is being dominated by some other objective vector in the same set.

Two other useful central concepts exist in multi-objective optimization. Namely, the ideal point and the nadir point. The ideal point $\mathbf{z}^*$ is defined as the objective vector for representing the lower bounds of a Pareto optimal set. The components of the ideal point can be calculated by optimizing each of the objectives in a multi-objective optimization problem independently. The nadir point $\mathbf{z}^{\text{nad}}$ is defined as the upper bounds of the Pareto optimal set. Calculating the nadir point is much more difficult than calculating the ideal point, because it depends on the Pareto optimal set and in the general case the full extent of the set is not known. A technique for calculating the ideal point and a representation of the nadir point is discussed later in Section 2.3.4.

It is also beneficial to note that $\mathbf{z}^* \notin Z$, and $\mathbf{z}^{\text{nad}} \in_{\text{maybe}} Z$; the ideal point is not the image of

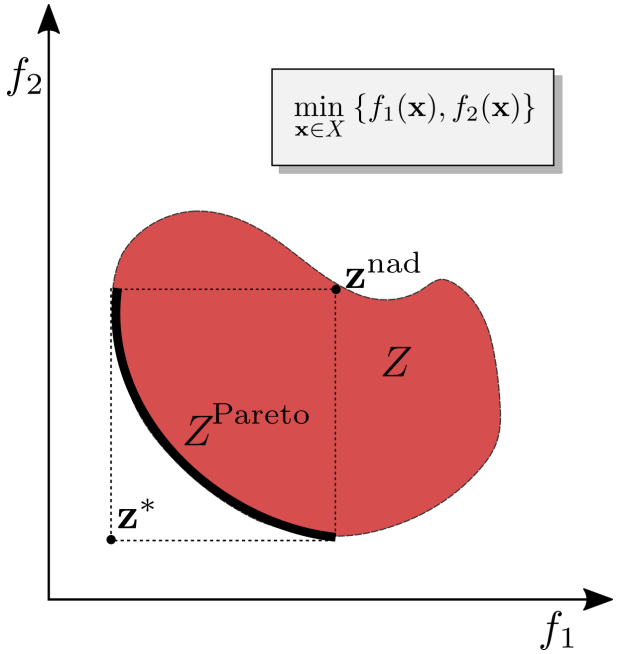


Figure 1: *The central concepts of multi-objective optimization conceptualized graphically for a problem with two objectives. The image of the feasible set Z (filled area), the Pareto optimal set Z^{Pareto} (the thick line segment), the ideal point $\mathbf{z}^*$, and the nadir point $\mathbf{z}^{nad}$ are pictured.*

a feasible solution[4], and the nadir point may or may not be the image of a feasible solution. The feasibility of the nadir point depends on the properties of Z, such as its concavity and connectedness.

The central concepts presented in this section have been conceptualized visually in Figure 1 for a multi-objective optimization problem with two objective functions to be minimized. It is evident from Figure 1 how the feasibility of the nadir point depends on Z and not just Z^{Pareto}; the feasible set can be trivially transformed to not include the nadir point without affecting the visualized Pareto optimal set.

[4]If it was, the multi-objective problem would have an unambiguous optimal solution.

2.3 Solving multi-objective optimization problems

Multi-objective optimization problems are not straightforward to solve, because the solution is ambiguous. Without any kind of preference information, it is impossible in a Pareto optimal set to say that one solution is better than the other. The number of possible Pareto optimal solutions can also be very large, even uncountable, which poses its own challenges: how many Pareto optimal solutions should be calculated? Should the calculated representation of the Pareto optimal set be evenly distributed? Is some region of the Pareto optimal set more preferable than the others? This section presents methods for solving multi-objective optimization problems by means of scalarization. A brief overview of other existing methods is also given at the end of this section.

2.3.1 A word on solving single-objective optimization problems

A common way of solving multi-objective optimization problems consists of solving single-objective optimization problems representing the original multi-objective optimization problem, combined with preference information given by a decision maker. Therefore, this subsection starts with a brief discussion on how to solve single-objective optimization problems. The concept of a decision maker and preference is discussed in Section 2.4.

To solve a single-objective optimization problem, it is necessary to find a solution $\mathbf{x} \in X$ that optimizes either (2.6) or (2.7). Various methods exist in literature to solve for $\mathbf{x}$. Two generally accepted and known methods for solving single-objective optimization problems are the simplex method and Newton's method. The simplex method transforms a linear single-objective optimization problem into a system of linear equations, which can then be solved using matrix operations. On the other hand, Newton's method does not require the objective function to be linear and it uses the derivatives of the objective function to iteratively improve the solution of the problem. However, Newton's method requires a good initial guess of the solution to the single-objective optimization problem to work well, which also means it is a local method – it will find the optimal solution close to the initial guess, which means the found solution is not necessarily a global solution. The simplex method does not require an initial guess and is able to find the global optimum. Moreover, the time complexity of

the simplex method is generally greater when compared to Newton's method, assuming that Newton's method converges fast enough to a local solution. The simplex method and Newton's method are both – and accompanied by many more similar methods – presented in [25, Chapter 9 and 10].

However, two methods important in this thesis for solving single-objective optimization problems are needed in particular: sequential least squares programming (SLSQP) [26, Chapter 10] and Coin-or branch and cut (CBC) [27]. SLSQP is able to solve non-linear constrained single-objective optimization problems, but requires a good initial guess of the solution to perform well which makes it a local method akin to Newton's method. SLSQP also requires the objective function and the constraint functions to be differentiable. When training a belief rule-based system discussed in Section 3.6, a non-linear constrained single-objective optimization problem emerges which can be solved for example utilizing SLSQP.

CBC is suitable for solving mixed integer problems: problems were the decision variable vectors and objective vectors have a mix of real and integer valued elements. Therefore, CBC is suitable for solving the formulated problem for computing the representation of the Pareto optimal set in the Finnish forestation problem discussed in the case study in Chapter 5.

This subsection will not delve further into the details of each method. For the rest of this section, it is assumed that a method for solving constrained non-linear single-objective optimization problems exists.

2.3.2 Scalarization

The vector valued objective function present in the definition of multi-objective optimization problems (2.9), can be transformed into a scalar valued function using a scalarizing function transformation:

$$\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m \xrightarrow{\text{scalarization}} \tilde{f} : \mathbb{R}^n \to \mathbb{R}. \tag{2.12}$$

The resulting single-objective optimization problem, when a scalarizing function transformation (2.12) is used on the objective function of a multi-objective optimization problem,

can then be solved using the methods discussed in Section 2.3.1. However, for the solution to be Pareto optimal the scalarization used in (2.12) must be congruent to certain conditions, which are defined for example in [24, Chapter 3]. For the rest of this thesis, it is assumed that the function used for scalarization guarantees the solution to the resulting scalarized function to be Pareto optimal.

As an example of scalarization, the weighting method can be used to transform a multi-objective optimization problem (2.9) into the single-objective optimization problem

$$\min_{\mathbf{x}\in X} \sum_{i=1}^{m} w_i f_i(\mathbf{x}) \,, \quad \text{where } \sum_{i=1}^{m} w_i = 1 \text{ and } w_i \geq 0,\ i \in [1,m]. \tag{2.13}$$

In problem (2.13), w_i is the weight associated to the ith objective function. It is proven in [1, Chapter 3.1], that the the solution to (2.13) is Pareto optimal if the weights w_i are all positive and the solution found is unique.

The weighting method is able to produce different Pareto optimal solutions using a different choice of weights. However, there are limitations to the extent of its ability to produce Pareto optimal solutions. If the underlying Pareto optimal set is non-convex, the weighting method is not able to find all the Pareto optimal solutions [1, Chapter 3.1]. Therefore, the main merit of the weighting method is in its simplicity. An overview of other existing scalarizing functions can be found in [1] and [28], for example.

2.3.3 Achievement scalarizing functions

Achievement scalarizing functions are a particular set of functions to scalarize multi-objective optimization problems. An achievement scalarizing function is defined as

$$s_{\bar{\mathbf{z}}} : \mathbb{R}^m \to \mathbb{R} :\ s_{\bar{\mathbf{z}}}(\mathbf{z}) = s_{\bar{\mathbf{z}}}(\mathbf{f}(\mathbf{x})), \tag{2.14}$$

where $\bar{\mathbf{z}} \in \mathbb{R}^m$ is a vector in the objective space called a reference point, and $\mathbf{f}$ is the objective function of a multi-objective optimization problem (2.9). Using an achievement scalarizing

function, an achievement problem

$$\min_{\mathbf{z}\in Z} s_{\bar{\mathbf{z}}}(\mathbf{z}) \tag{2.15}$$

can be formulated. It is proven in [1, Chapter 3.5], that when (2.14) is increasing and the solution to the achievement problem is unique, then the solution is also Pareto optimal. For solving (2.15), it is required to find an objective vector $\mathbf{z}$ which minimizes it. Methods discussed in Section 2.3 are suitable for solving achievement problems as well.

In the case study discussed in Chapter 5, a representation of the Pareto optimal set is computed using an achievement scalarizing function – the guess function – originally used in the GUESS method [29]

$$s_{\bar{\mathbf{z}}}(\mathbf{z}) = \max_{i\in[1,m]} \left[\frac{z_i - z_i^{\text{nad}}}{z_i^{\text{nad}} - \bar{z}_i} \right], \tag{2.16}$$

where z_i, z_i^{nad}, and $\bar{z}_i$, are the components of the objective vector $\mathbf{z}$, nadir point $\mathbf{z}^{\text{nad}}$, and reference point $\bar{\mathbf{z}}$, respectively. Intuitively, the guess function, when minimized, results in a Pareto optimal vector $\mathbf{z}$, which has components z_i that have the minimized maximum deviation compared to the components of the nadir point. The vector $\mathbf{z}$ found resides on a line segment connecting the reference point and the nadir point. This is a very straightforward way of calculating a Pareto optimal solution, with minimal assumptions made. Only the nadir point of the original multi-objective optimization problem is required and the reference point chosen must be smaller[5] in every component compared to the nadir point. In [1, Chapter 5.7], it is shown that any Pareto optimal solution can be found utilizing (2.16). That is to say, the function (2.16) does not suffer from the same limitations as the weighting method (2.13) mentioned in Section 2.3.2 regarding the convexity of the Pareto optimal set.

The function in (2.16) can be used to generate a representation of a Pareto optimal set for a multi-objective optimization problem by generating a set of reference points and solving the resulting achievement scalarizing problem (2.15) for each generated point. This method of solving for a representation of the Pareto optimal set is somewhat similar to the

[5] Or bigger, if maximization is desired in the underlying multi-objective optimization problem.

normal-boundary intersection method [30], where a set of reference points is also generated. However, normal-boundary intersection might produce non-Pareto optimal solutions.

One more advantage achievement scalarizing functions have, is that an augmentation term can be used in the achievement scalarizing function formulation to guarantee Pareto optimality. An augmentation term could be for example the complement of the euclidean distance of the objective vector $\mathbf{z}$ in (2.16) to the problems ideal point weighted by some scalar ρ between 0 and 1; e.g. $-\rho \sum_{i=1}^{m} |z_i^* - z_i|$ where z_i^* are the components of the problem's ideal point. Examples of other available achievement scalarization functions, and their comparisons, can be found in [31].

2.3.4 Calculating the ideal and nadir points: the payoff table method

Computing the ideal point is simple: each objective function in a multi-objective optimization problem is optimized independently of the other objective functions. To illustrate this idea, consider the following example: suppose the ideal point is to be calculated for a multi-objective optimization problem defined in (2.9) with three objective ($m = 3$). Optimizing the first objective function f_1 will result in a value z_1^* which will be the first component of the ideal point $\mathbf{z}^*$. Similarly, z_2^* and z_3^* can be calculated by independently optimizing f_2 and f_3, respectively, and consequently leading to a complete formulation of the ideal point. The given example can be trivially generalized to any multi-objective optimization problem with an arbitrary number of objective functions.

Again, consider the example discussed in the previous paragraph with three objectives. Let $\mathbf{x}_i$ be the optimal solution when f_i is optimized independently and let $z_i^j = f_j(\mathbf{x}_i)$. In other words, z_i^j is the value of the jth objective function evaluated with the optimal solution found for the ith objective function f_1 when it is optimized independently. If the values z_i^j are arranged in a matrix with i being the row index and j the being column index, the result will

be

$$\begin{bmatrix} z_1^1 & z_1^2 & z_1^3 \\ z_2^1 & z_2^2 & z_2^3 \\ z_3^1 & z_3^2 & z_3^3 \end{bmatrix} \tag{2.17}$$

which is known as a payoff table. It is important to notice that the diagonal of (2.17) is equivalent to the ideal point, that is $\mathbf{z}^* = \{ z_1^1, z_2^2, z_3^3 \}$. Now, taking the worst objective value from each of the columns in (2.17) will give an approximation of the nadir point $\mathbf{z}^{\text{nadir}}$. This way of calculating the nadir point is known at the *payoff table method*.

As said, acquiring a representation of the nadir point using a payoff table does not necessarily result in a nadir point with components reflective of the true nadir point of the problem. The components may be too low, or too high, and using a nadir point produced by the payoff table method may result in inaccurate limits of the Pareto optimal set. In general, the true nadir point is hard to calculate for a multi-objective optimization problem. This is why methods which approximate the nadir points are used instead, and the payoff table method's simplicity makes it an attractive option. Moreover, in the case study discussed in Chapter 5 the accurate limits of the attainable objective vector values are not of critical importance, and therefore the payoff table method is a suitable way for calculating the estimate of the nadir point in the case study.

An alternative way, which is arguably a better but slower way for computing an estimate of the nadir point compared to the payoff table method can be found in [32], where a method based on evolutionary algorithms is presented. Similarly in [33] a method based on a mix of evolutionary algorithms and local search methods is discussed.

2.4 The decision maker and preference

So far, only the optimization aspect of multi-objective optimization has been discussed. This section presents the concept of a decision maker, what is meant by their preferences, and how the preferences of a decision maker can be modelled. These concepts are in a central role in multi-objective optimization.

2.4.1 The decision maker

In the context of multi-objective optimization, the decision maker is usually a human, who is an expert in the field related to a problem being solved. The decision maker can be a single human, or a group of humans. When a group of decision makers is in question, the decision making process is known as group decision making. The decision maker can also be artificial, for example an expert system or an artificial intelligence. However in this thesis, only the case of a single human decision maker is considered.

The role of the decision maker is that of providing preference information, which is not considered in the general formulation of multi-objective optimization problems (2.9). Preference information can be given by the decision maker in different forms and at different times, when solving a multi-objective optimization problem.

Preferences may consist of weights given individually for objectives, for example. The weights given can be used for instance in (2.13) to solve a multi-objective optimization problem. Another form of preference information is aspiration levels, which are acceptable or desired values given by the decision maker for each objective function. The aspiration levels can be used to form a reference point: a vector consisting of desired values for each objective function according to the preferences of the decision maker. This kind of a reference point can be used in achievement scalarizing functions like (2.16), for example.

When preferences are given by a decision maker before a multi-objective optimization problem is solved, preferences are said to be given a priori. If preferences are given after an approximation of the Pareto optimal set has been computed, preferences are said to be given a posteriori. Preferences can also be given interactively, which means they are given during the process of solving a multi-objective optimization problem; the preferences are often used to guide the optimization process itself. Different methods for solving multi-objective optimization problems are usually classified using these three terms: a priori methods, a posteriori methods, and interactive methods [1]. Interactive methods will be discussed further in Section 2.5.

In this thesis, two assumptions are made regarding the decision maker. The first assumption is that a decision maker will always prefer a non-dominated objective vector to a dominated

one. Based on this assumption, it can be stated that a decision maker will be always interested only in solutions present in the Pareto optimal set of a problem, when the set or a representation of the set is available.

The second assumption is that if the decision maker prefers the objective vector $\mathbf{z}_1$ to the objective vector $\mathbf{z}_2$ and the objective vector $\mathbf{z}_2$ to $\mathbf{z}_3$, then it must be that he/she also prefers $\mathbf{z}_1$ to $\mathbf{z}_3$. More formally, this second assumption can be expressed as

$$\mathbf{z}_1 \succeq \mathbf{z}_2 \wedge \mathbf{z}_2 \succeq \mathbf{z}_3 \Rightarrow \mathbf{z}_1 \succeq \mathbf{z}_3, \tag{2.18}$$

where the notation "$\succeq$" indicates that the element on its left is preferred or equal in terms of preference, when compared to the element on its right. A strict preference would be indicated by "$\succ$" in a similar fashion.

A decision maker who abides to the two assumptions made, is considered to be a rational decision maker in this thesis. Henceforth, all decision makers discussed will be assumed to be rational decision makers.

2.4.2 Modelling preferences: the value function model

One way to model the preferences of a decision maker, is to assume a value function of the form

$$U : \mathbb{R}^m \rightarrow \mathbb{R} : U(\mathbf{z}) \tag{2.19}$$

to exist. The function (2.19) is used to give a value reflecting the decision maker's preferences for objective vectors $\mathbf{z}$. If for two objective vectors $\mathbf{z}_1$ and $\mathbf{z}_2, U(\mathbf{z}_1) > U(\mathbf{z}_2)$, then $\mathbf{z}_1$ is preferred to $\mathbf{z}_2$ according to the decision maker's preferences. The objective vectors in an objective vector set can be at least partially ranked according to the preferences of a decision maker using a value function.

Because of the assumption stating the decision maker being rational, the value function can be assumed to be monotonic. In other words, by setting all the objective values to some constant and decreasing only one of the objective values, the output of the value function

should be monotonically increasing. This idea can be expressed as the conditional statement

$$\bigvee_{i=1}^{m} z_i \leq z'_i \Rightarrow U(\mathbf{z}) \geq U(\mathbf{z}') \quad \forall\, \mathbf{z}, \mathbf{z}' \in Z. \tag{2.20}$$

While in (2.20) the objective vectors are limited to the vectors present in the image of the feasible set Z, it is important to note that the condition is valid for any real-valued vectors in any m-dimensional space.

Obtaining the value function in (2.19) for a decision maker is not simple. It is unrealistic for a decision maker to be able to directly tell what the value function reflecting his/her preferences is. Therefore, the value function is usually inferred in some way using auxiliary indicators of the decision maker's preferences, such as is done in pair-wise comparisons where the decision maker is asked which one of two objective vectors presented he/she prefers.

For example in [21], a value function is inferred from simple pair-wise comparisons of two objective vectors, which are presented to the decision maker at regular intervals. The decision maker is asked which objective vector they prefer or if the vectors are equally preferred. Based on the information gained from these pair-wise comparisons, a value function is inferred.

If a value function is inferred, it can be optimized and the resulting optimum will result in the best solution according to the decision maker's preferences. The value function can also be used to drastically decrease the computational complexity of large multi-objective optimization problems by transforming the problem into a single-objective optimization problem. In some cases, a partially inferred value function can be used to produce new non-dominated solutions, which were not present in a previously computed representation of a Pareto optimal set to guide the optimization process toward solutions of greater interest.

2.5 Interactive methods

As was discussed in Section 2.4.1, one category of methods for solving multi-objective optimization problems is interactive methods. Real-life optimization problems can often be

computationally very expensive with many decision variables and objective vectors. A computed representation of the Pareto optimal set might be exceptionally large, even uncountable, and the decision maker's familiarity with the available Pareto optimal solutions may be very limited. This means that the relevant subset of the Pareto optimal solution set, which contains the most interesting objective vectors for the decision maker can be assumed to be unknown before any preference information is gained.

Therefore, a priori and a posteriori methods can be very taxing – either for the decision maker or computationally. Interactive methods aim to tackle these issues by incorporating the decision maker's preferences in the optimization process itself. For example, the preference information can be used to reduce the search space for feasible Pareto optimal solutions. Interactive methods also allow the decision maker to learn more about the underlying problem by providing the decision maker with intermediate solutions and additional information, like upper and lower bounds of the available feasible solutions.

A recent review on existing interactive multi-objective optimization methods can be found in [34]. In what follows, two interactive methods for solving multi-objective optimization problems are presented and briefly discussed. The methods have been chosen to conceptualize the idea of how much interactive methods can differ from each other in how they elicit the preferences of a decision maker and in their approach to find the best solution.

In the E-NAUTILUS [35] method, the optimization process starts by showing the decision maker the nadir and ideal points of the multi-objective optimization problem being solved. The decision maker is then shown a few objective vectors, which are all strictly better than the nadir point and mutually non-dominated. Each shown objective vector is also associated with limits of the best reachable Pareto optimal solution from that point. The decision maker then chooses one of the intermediate vectors to be his/her most preferred and that objective vector is then used again to generate a new set of intermediate objective vectors, which are all strictly better than the previously chosen vector and mutually non-dominated. The decision maker is then asked once again to choose his/her most preferred objective vector from the intermediate vectors. This iteration process is continued until a vector in the Pareto optimal set is reached. The central idea of E-NAUTILUS is to start from the worst possible objective vector and gradually improve upon it. The decision maker is therefore never asked to make

trade-offs. Instead he/she can always improve upon the previously chosen objective vector. The authors of the E-NAUTILUS method argue that this kind of process is able to reduce the anchoring effect[6] in a decision maker in a decision making process.

The idea behind the Synchronous NIMBUS[7] [3] method is to show the decision maker a Pareto optimal objective vector and to ask the him/her for preference information related to the individual objective function values present in the vector. The decision maker is asked whether he/she would like an objective function's value to be improved from the shown value, improved from the shown value until some aspiration level is reached, accepted at their current level, allowed to be impaired until some bound is reached, or allowed to change freely. New Pareto optimal objective vectors are computed using the provided preference information in different types of achievement scalarizing functions. The decision maker can then choose his/her most preferred objective vector from the new computed vectors, or to save part of the computed objective vectors to an archive to be viewed later. The decision maker can also choose to see intermediate objective vectors between two computed vectors if they so wish. The process can be continued using the new objective vector selected by the decision maker and asking the decision maker for preference information on the new vector. This iteration process is continued until the decision maker is satisfied with a found objective vector. The central benefit of Synchronous NIMBUS is that it uses different kinds of scalarizing functions, which all produce different Pareto optimal objective vectors using the same preference information. Bias introduced by choosing just a single scalarizing function is therefore reduced.

[6] A type of cognitive bias where the decision maker is affected by some initial piece of information, affecting subsequent decisions.

[7] Synchronous NIMBUS is available to be used directly in a web browser at `https://wwwnimbus.it.jyu.fi/`

3 Belief rule-based systems

This chapter begins with a less formal introduction to traditional rules and rule sets given in Section 3.1. Following it, a detailed discussion on belief rule-based systems is given in Sections 3.2, 3.3, and 3.4. In Section 3.5, it is shown how a belief rule-based system can be built, and in Section 3.6 it is shown how the system can be adaptively trained. This chapter will conclude with Section 3.7, where a numerical example is given, and with Section 3.8, where a few existing practical applications applying belief-rule based systems in real-world problems are discussed. The formal presentation of belief-rule based systems given in this chapter is based on the RIMER methodology [14].

In an attempt to not inflate the amount of different symbols used and introduced in this thesis, symbols already presented in the previous chapter, Chapter 2, will be reintroduced in this chapter defined to convey a completely different meaning compared to what was discussed in the context of multi-objective optimization. This choice has also been made to keep the used symbols similar to the symbols presented in the original literature, on which the content of this chapter is partially based on. The reader is therefore advised to keep this detail in mind.

3.1 Rules and rule sets

Rules are a natural way of conveying information on how an agent or system should work according to its surrounding environment. Information from the surrounding environment is gathered via observations of attributes. A rule can be as simple as an if... then... statement; such a rule could be for example:

Rule 1: "*If it rains outside, bring an umbrella.*".

A rule consists of three elements: the precedent, the condition, and the consequent. Take Rule 1 as an example. In Rule 1, the precedent should answer the question implied by the condition "If it rains". Therefore, the precedent is tied to the observation of rain outside, and the precedent should answer the question "Does it rain outside?". This will result in an

observation producing an observed attribute, which will have the value *true* if it rains, or *false* if it does not rain. The consequent "Bring an umbrella." should follow the condition only when the answer implied by the precedent is true. When all the necessary observations have been made to produce all the needed attributes to answer the question implied by the precedent, and it has been established whether the consequent should follow the precedent, the rule is said to be resolved.

Still considering Rule 1, suppose there was an agent named Bob who is contemplating whether he should bring an umbrella outside or not. Bob needs to look outside and see if it rains or not; he needs to make an observation of his surroundings and gather information on relevant attributes. Using the observed attribute with the precedent and the condition, the rule can be resolved: if Bob observes rain, the attribute value tied to the condition will be *true*, and the rule resolves to the consequent: "Bring an umbrella.".

However, Rule 1 does not state anything regarding the case when it does not rain outside. One could argue that the rule implicitly suggests not to bring an umbrella in the absence of rain, but it is not an explicit consequent of the rule in the case of no rain. To cover the case of no rain, an additional rule should be stated:

Rule 2: "*If it does not rain outside, do not bring an umbrella.*".

In turn, Rule 2 does not state anything regarding the case when it does rain outside. For Bob to have a coherent set of actions to take, whether to bring an umbrella outside or not, Rules 1 and 2 should be combined into a single rule set:

Rule set 1: $\begin{cases} \text{“}\textit{If it rains outside, bring an umbrella.}\text{“} \\ \text{“}\textit{If it does not rain outside, do not bring an umbrella.}\text{“} \end{cases}$

Rule set 1 is now a complete rule set; a definite consequent follows all possible precedents, namely "It rains outside." and "It does not rain outside.". If there was a third rule with the precedent: "It might be raining outside", rule set 1 would be incomplete and is said to contain a degree of ignorance: it does not state what to do in the case when the observation of rain is indefinite.

That being said, in real-life observations are seldom definitive. Covering all possible cases of different observations made by Bob for Rule set 1 in real-life would mean an addition of an indefinite number of rules. In practice, trying to cover all cases is an exercise in futility in real-life situations.

Could the conditions of rules in a rule set have a degree of truthiness[1] associated to them, instead of being simply true of false? Could such a rule set cover a continuous range of observations and outcomes? Is it possible to infer an appropriate consequent from such a rule set, and how to interpret the outcome? This is where belief rule-based systems come into play.

3.2 Belief rules

In a belief rule R, the precedent is defined as a set of finite arrays of referential values. Taking Rule 1, and adding to the conditional the clause "...and the ground looks wet.", results in:

Rule 3: "*If it rains outside and the ground looks wet, bring an umbrella.*".

The condition in Rule 3 now implies two distinct questions: "Does it rain outside?" and "Does the ground look wet?". Both questions can have an answer of *true* or *false*. To formulate a consequent capable to answer both questions, the consequent is defined as a set of arrays A, also called the packet precedent. The set A consists of arrays of finite values A_i, representing the possible attribute values that can result from an observation tied to each precedent. In case of Rule 3, and representing *false* by 0, and *true* by 1, the set A is defined as

$$A = \{A_1, A_2\} = \{\{0,1\},\{0,1\}\}, \tag{3.1}$$

where A_1 consists of all the possible values of the attribute resulting from the observation "Does it rain outside?", and A_2 consists similarly of the possible values for the attribute resulting from the observation "Does the ground look wet?".

[1] A degree indicating how true something is.

The arrays in (3.1) are said to contain the referential values for each of the attributes relevant to the rule. If the attribute resulting from the observation "Does it rain outside?" is labelled as x_1, and the attribute resulting from the observation "Does the ground look wet?" is labelled as x_2, then the attributes x_1 and x_2 can take values only present in the arrays A_1 and A_2 respectively, that is $x_1 \in A_1$ and $x_2 \in A_2$.

Following the RIMER approach [14] to define a belief rule, let A consist of T arrays of finite values A_i, where the index i refers to the ith element in the set A. Let each array A_i then consist of $n = |A_i|$ values, each value assumed to be real valued. Then, $A_{i,n}$ will refer to the nth value in the ith array A_i in A. The packet precedent can then be defined as

$$A = \left\{ A_i = \left\{ A_{i,j} \mid j \in [1, |A_i|] \right\} \mid i \in [1, T] \right\}, \quad \text{with} \quad i, n, T \in \mathbb{N}. \tag{3.2}$$

Similarly, an array of finite values denoted by D is defined, and it contains the referential values for the consequent in a belief rule. The array D consists of all the possible values the consequent can take. In Rule 3, the consequent can be either *false* or *true*. Using the same representation for true and false, like mentioned before, the possible values for the consequent will be $\{ 0, 1 \}$. Generally, in a belief rule the consequent with N possible referential values is defined as

$$D = \{ D_i \mid i \in [1, N] \}, \quad \text{where} \quad i, N \in \mathbb{N}. \tag{3.3}$$

where D_i refers to the ith component of D.

The input to a belief rule is also defined as an array. Looking back at Rule 3, and the attribute values labelled as x_1 and x_2, the input to Rule 3 is the vector $X = \{ x_1, x_2 \}$. The input can be generalized as

$$X = \{ x_i \mid i \in [1, T] \}, \quad \text{where} \quad i \in \mathbb{N}, \tag{3.4}$$

where x_i refers to the ith element in X. Attributes x_i in (3.4) can take on values equal to the values defined in the finite arrays A_i present in the packet precedent A in (3.2).

The condition of a belief rule can be defined as a logical relationship F between the input X and the packet precedent A as

$$F : x_1 \text{ is } A_1^* \wedge x_2 \text{ is } A_2^* \wedge \cdots \wedge x_T \text{ is } A_T^*, \tag{3.5}$$

where $A_i^* \in A_i$ represents the referential value taken by the attribute x_i in the belief rule. The logical relation 'and': $\wedge$, present in (3.5), could be replaced by some other logical relation, but in this thesis the RIMER approach is followed, which uses the $\wedge$ relation.

Assuming multiple belief rules to exists, and labeling the rules as R_k, with R_k referring to the kth rule, the consequent D of each belief rule is associated with an array of real valued belief degrees $\beta_{i,k}$, where the indices i,k refer to the ith belief degree in rule k. The belief degrees are used to associate a degree of truthiness to each element in the consequent D. That is, $\beta_{i,k}$ is used to associate a degree of truthiness to the ith element in D to be the consequent of the kth rule. Additionally, θ_k is defined as a real valued scalar, a weight for the kth rule, and $\delta_{i,k}$ is defined to be an array of real values representing the attribute weights for each of the attributes to the input of rule k, with the index i representing the ith attribute's weight in $\delta_{i,k}$. Using A^k to denote a subset of A, and T_k as the number of finite vectors in A^k, a kth belief rule is then defined as

$$\begin{aligned} & \text{IF } x_1 \text{ is } A_1^k \wedge x_2 \text{ is } A_2^k \wedge \ldots \wedge x_{T_k} \text{ is } A_{T_k}^k, \\ R_k: \quad & \text{THEN } \left\{ \left(D_i, \beta_{i,k}\right) \mid i \in [1,N] \right\}, \\ & \text{with an associated rule weight } \theta_k, \\ & \text{and attribute weights } \left\{ \delta_{i,k} \mid i \in [1,T_k] \right\}, \end{aligned} \tag{3.6}$$

with the following property for the belief degrees

$$\sum_{i=1}^{N} \beta_{i,k} \leq 1. \tag{3.7}$$

In (3.6), it is understood that the rule R_k has its own packet precedent $A^k \subset A$, with T_k referential value vectors defined. The rule weight θ_k represents how important the rule R_k

is compared to other rules. The attribute weights $\delta_{i,k}$ represent how important attributes are compared to each other in the rule R_k. The rule weights and attribute weights will be discussed later.

The output of a belief rule is a belief distribution. The equality of the property (3.7) is valid when the output of a rule can be fully expressed using the consequent's referential values D_i. Otherwise, a degree of ignorance – meaning that the output of a belief rule cannot be fully specified using the consequent D – is present in the output of the rule.

By constructing multiple rules like defined in (3.6), a belief rule base consisting of L belief rules, can be established and is represented as

$$R = \langle X, A, D, F \rangle . \tag{3.8}$$

Inference can then be performed on the belief rule base R for a specific input vector X. How to construct an input vector based on an observation, and how inference can be performed on a belief rule set, will be discussed in the following sections.

3.3 Input transformation

Looking back at the example in Section 3.1, Bob may not be sure if it rains outside: Bob might believe that it is raining outside with a 90% probability. If the available attributes for the input were *false* and *true*, Bob's observation cannot be fully expressed by these values. Like discussed previously, it would be futile to redefine Rule set 1 to cover all possible cases – which would be analogous to expanding each vector in the packet precedent of a belief rule set to include all possible values for the observed attributes. This is why a mechanism for mapping observed attributes to available referential values for said attributes is defined.

Because observations may not always result in attributes that can be specified using the referential values for the precedents defined in the packet precedent (3.2), some mechanism must be defined to map observed attributes to the available referential values. An observation H is a vector consisting of elements h_i, where the observed attribute h_i is associated to the

ith input attribute to a belief rule, x_i. Formally

$$H = \{\, h_i \mid i \in [1,T] \,\}. \tag{3.9}$$

For a single observed attribute h_i and the precedent referential values A_i associated to the ith input attribute to a belief rule, the observed attribute is transformed into a belief distribution consisting of the elements $\alpha_{i,n}$. Each $\alpha_{i,n}$ conveys a degree of truthiness that the observed attribute h_i maps to the referential value $A_{i,n}$ in the packet precedent A. The belief distribution for the ith observed attribute is defined as

$$s_i = \{\, (A_{i,n}, \alpha_{i,n}) \mid n \in [1, |A_i|] \,\}, \tag{3.10}$$

The belief degrees $\alpha_{i,n}$ present in (3.10) associated to an observed attribute value h_i can be calculated as follows [36]:

$$\alpha_{i,n} = \left\{ \min\left(\max\left(\frac{A_{i,j+1} - h_i}{A_{i,j+1} - A_{i,j}}, 0 \right), \max\left(\frac{h_i - A_{i,j-1}}{A_{i,j} - A_{i,j-1}}, 0 \right) \right) \mid j \in [1, |A_i|] \right\}, \tag{3.11}$$

where $A_{i,0} = A_{i,T}$ and $A_{i,|A_i|} = A_{i,1}$. The belief degrees calculated using (3.11) have the property

$$\sum_{j=1}^{|A_i|} \alpha_{i,j} \leq 1. \tag{3.12}$$

The equality in property (3.12) is valid when an observed attribute h_i can be fully expressed by the values defined in the precedent referential values A_i. If the equality in (3.12) is not valid, a degree of ignorance is present; the attribute h_i cannot be fully specified using the referential values in A_i.

For multiple inputs, the belief distribution in (3.10) can be generalized to a set of belief

distributions

$$S' = \{ \{ (A_{i,n}, \alpha_{i,n}) \mid n \in [1, |A_i|] \} \mid i \in [1, T] \}. \tag{3.13}$$

3.4 Inference mechanism

So far, the concept of a belief rule, and how to transform an observed attribute into an input attribute to a belief rule, has been defined in Sections 3.2 and 3.3. To really make use of a belief rule base, an inference mechanism is needed.

For a belief rule base consisting of L rules, the rule weights θ_k, the belief degrees $\beta_{i,k}$, and the attribute weights $\delta_{i,k}$, are assumed to be known.[2] An activation weight w_k, for the kth rule can then be calculated as follows [36]:

$$w_k = \frac{\theta_k \prod_{i=1}^{T_k} \left(\alpha_{i,n}^k \right)^{\bar{\delta}_{i,k}}}{\sum_{l=1}^{L} \left[\theta_l \prod_{i=1}^{T_l} \left(\alpha_{i,n}^l \right)^{\bar{\delta}_{i,l}} \right]}, \tag{3.14}$$

where

$$\bar{\delta}_{i,k} = \frac{\delta_{i,k}}{\max_{i \in T_k} \left(\delta_{i,k} \right)}. \tag{3.15}$$

In (3.14) the term $\alpha_{i,n}^k$ indicates how the ith attribute in an input to the kth rule in a belief rule base matches the precedent value $A_{i,n}^k$ defined in the rule. The product in the nominator of (3.14) is an aggregation of the belief degrees defined in (3.13) for each attribute x_i present in the input of the rule. Since the logical relation of the precedents in a belief rule is assumed to be the 'and' connective, $\wedge$, the belief degrees are combined using a product. In other words, a belief rule k is active, meaning it has a non-zero activation weight, only when all of the attributes in the input to the rule can be associated to the precedent referential values

[2] For example, if the rule base is used to model the functioning of some system in a specific domain, a corresponding domain expert can give estimates for the attribute weights.

$A_1^k, A_2^k, \ldots, A_{T_k}^k$ in the condition of the rule. Otherwise, the rule activation weight w_k will be zero, and the rule is not active for the given input.

Moreover, the importance of each attribute in the kth belief rule is indicated by the attribute weights $\delta_{i,k}$, which are normalized to $\bar{\delta}_{i,k}$ according to (3.15). Thus, in the kth rule an attribute with a normalized attribute weight of 1 is deemed the most important affecting the activation weight in (3.14) substantially, while an attribute with an attribute weight of 0 has no impact on the activation weight. This is a direct consequent of the nature of the exponential function present in the multiplicands in (3.14).

Lastly, the rule weights θ_k are used to represent the relative importance of each rule in a belief rule set, meaning that the rules with the highest rule weights are the most important ones and the rules with the lowest weights the least important. In this thesis, rule weights are defined to be positive and normalized such that they sum to unity:

$$\sum_{k=1}^{L} \theta_k = 1. \tag{3.16}$$

After calculating the rule weights, inference can be finally conducted on the rule base. The mechanism to conduct the inference used in this thesis is based on the RIMER methodology presented in [14]. RIMER itself is based on Dempster-Shafer theory, fuzzy set theory, and decision theory, each discussed in [36]. This thesis will not dive further into the details of the inference mechanism and will simply present relevant results.

To aggregate the outputs of the different belief rules in a rule base, a combined belief degree β_n is calculated to produce a final belief distribution

$$S = \{ (D_n, \beta_n) \mid n \in [1, N] \} \tag{3.17}$$

on the referential values D, representing the output of a belief rule-base. A real value can be then associated to each consequent referential value in D – that is, for each $D_i,\ i \in [1, N]$ –

using a utility function defined as

$$u : D_i \to \mathbb{R}. \tag{3.18}$$

The utility function (3.18) is specific to the domain of application the belief rule base is associated to and it is reflective of how relevant each of the components D_i are to the problem at hand.

After the utility function u is defined, a numerical output y associated to the belief distribution (3.17) can be defined to be

$$y = \sum_{n=1}^{N} u(D_n)\beta_n. \tag{3.19}$$

According to [36], and assuming that the rules in a rule base are independent of each other, the combined belief degrees for each consequent referential value can be calculated as follows:

$$\beta_n = \frac{\prod_{k=1}^{L}(w_k\beta_{n,k} + 1 - w_k\sum_{i=1}^{N}\beta_{i,k}) - \prod_{k=1}^{L}(1 - w_k\sum_{i=1}^{N}\beta_{i,k})}{\sum_{j=1}^{N}\prod_{k=1}^{L}(w_k\beta_{j,k} + 1 - w_k\sum_{i=1}^{N}\beta_{i,k}) - (N-1)\prod_{k=1}^{L}(1 - w_k\sum_{i=1}^{N}\beta_{i,k}) - \prod_{k=1}^{L}(1 - w_k)}, \tag{3.20}$$

where $n \in [1, N]$ refers to the nth consequent in D. It is evident in (3.20) that when a rule k is not activated – meaning its activation weight w_k is zero – the combined belief β_n will not be affected by rule k. Furthermore in (3.20), if a rule k is active and the rule has an individual belief degree $\beta_{n,k}$ associated to the output D_n in the rule, the combined belief degree β_n associated to D_n must be non-zero. The magnitude of the degree of a rule k affecting the combined belief degree is affected both by the importance of the kth rule to the overall output, indicated by the rule weight θ_k, and by the degree of activation of the precedent in the kth rule by an input vector, which is indicated by the activation weight w_k.

Symbol	Meaning
A	packet precedent
A_i	a vector of real values representing the referential values for the i precedent
$A_{i,j}$	the j referential value for the ith precedent
D	a vector of real values representing the referential values for the consequent
D_i	the ith referential value for the consequent
R	a belief rule base
R_k	the kth belief rule of a belief rule base
X	the input to a belief rule
x_i	the ith attribute value in the input X
F	the logical mapping between an input and the packet precedent in a belief rule
A_i^k	the precedent referential value taken by the the attribute x_i in rule R_k
$\beta_{i,k}$	belief degree associated to rule R_k and consequent referential value D_i
β_n	the combined belief degrees associated to D
θ_k	the rule weight associated to rule R_k
$\delta_{i,k}$	the attribute weight associated to attribute x_i in rule R_k
$\bar{\delta}_{i,k}$	the normalized attribute weight associated to attribute x_i in rule R_k
H	a vector consisting of observed attributes
h_i	the ith observed attribute ion H
s_i	the belief distribution associated to h_i
$\alpha_{i,n}$	a vector of belief degrees associated to h_i
w_k	the activation weight associated to rule R_k
S'	the set of belief distribution associated to H and each vector $\alpha_{i,n}$
S	the set of belief distribution associated to D and each β_n
u	an utility function
y	the numerical output aggregated from S using u

Table 1: *List of symbols used for defining belief rule-based systems. Symbols with the sub-index n are to be regarded as vectors consisting of real valued elements. See the text for a more detailed description for each of the symbols.*

3.5 Building belief rule-based systems

In the previous sections various concepts like a belief rule base, input transformation, and an inference mechanism have been introduced. A belief-rule based (BRB) system is a system that combines those concepts. In Figure 2, a schematic representation of a BRB system is given illustrating its various components, and how the components depend on each other through different parameters. In essence, BRB systems model relationships between precedent and consequent attributes given an input based on an observation. A summary of the different symbols introduced is given in Table 1.

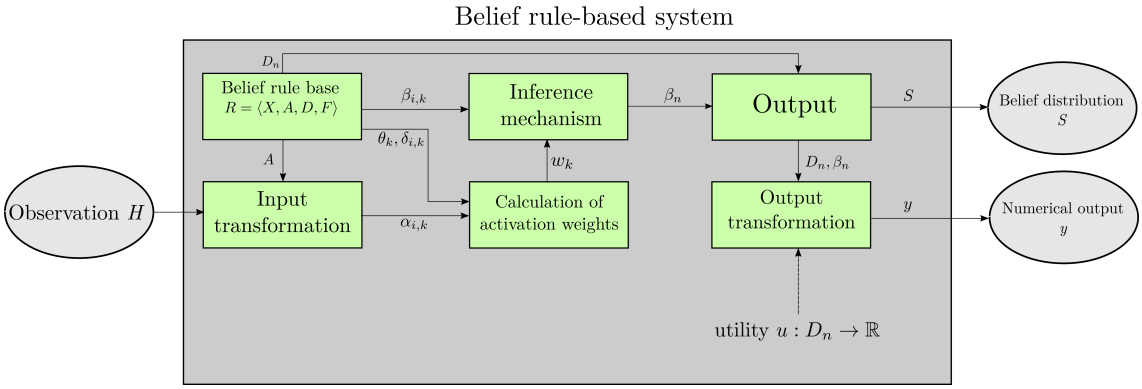


Figure 2: *A schematic representation of the components of a BRB system. The arrows represent the dependencies between the components.*

To build a BRB system, an initial[3] rule base must be build. This is best demonstrated using a simple example.

Example 1: Consider a BRB system, which models the relationship of an input X of two numerical attributes x_1 and x_2 to some numerical output y, which in turn will be represented by three consequent referential values $D = \{D_1, D_2, D_3\}$. The packet precedent A of such a system has two components, namely A_1 and A_2. Furthermore, suppose that the components A_1 and A_2 contain both three referential values, that is $A_1 = \{A_{1,1}, A_{1,2}, A_{1,3}\}$ and $A_2 = \{A_{2,1}, A_{2,2}, A_{2,3}\}$. From these precedents, a total of nine belief rules can be constructed. The nine rules cover all the cases of possible value pairs of the input $X = (x_1, x_2)$, where the attributes x_1 and x_2 take on values available in the vectors A_1 and A_2, respectively. An example of such a belief rule according to (3.6) is

[3]Before any training is done.

$$
\begin{aligned}
& \text{IF } x_1 \text{ is } A_1^1 \ \wedge \ x_2 \text{ is } A_2^1 \,, \\
R_1: \quad & \text{THEN } \left\{ \, (D_i, \beta_{i,1}) \mid i \in [1,3] \, \right\}, \\
& \text{with an associated rule weight } \theta_1, \\
& \text{and attribute weights } \left\{ \, \delta_{i,1} \mid i \in [1,2] \, \right\}.
\end{aligned}
\tag{3.21}
$$

The possible combinations of the precedent referential values present in the precedent of the rule can be calculated using the Cartesian product over the two referential value vectors A_1 and A_2:

$$
A_1 \otimes A_2 = \left\{ \, (a_1, a_2) \mid a_1 \in A_1 \text{ and } a_2 \in A_2 \, \right\}, \tag{3.22}
$$

which results in nine different combinations.

The Cartesian product (3.22) can be generalized for any packet precedent with any number of referential value vectors T as

$$
\bigotimes_{i=1}^{T} A_i = \left\{ \, (a_1, a_2, \ldots, a_T) \mid a_1 \in A_1 \text{ and } a_2 \in A_2 \ldots \text{ and } a_T \in A_T \, \right\}. \tag{3.23}
$$

The number of rules L in a rule base, with the packet precedent A, is then the total number of combinations in the Cartesian product (3.23), or equivalently

$$
L = \prod_{i=1}^{T} |A_i| \,. \tag{3.24}
$$

Returning to Example 1, using (3.24) with the defined vectors A_1 and A_2, the total number of rules amounts to $L = 9$, just like previously stated.

The belief degrees $\beta_{i,k}$, rule weights θ_k, and attribute weights $\delta_{i,k}$ are yet to be defined. The belief degrees $\beta_{i,k}$ can be assigned by an expert with expertise on the system being modelled by the BRB system, or be inferred from existing data. For example, suppose the BRB system

in Example 1 is used to model the function

$$f : \mathbb{R}^2 \to \mathbb{R};\ f(x_1, x_2) = x_1 + x_2, \tag{3.25}$$

that is, the goal is to construct a BRB system able to map the input variables x_1 and x_2 to the numerical output $x_1 + x_2$.

Using f defined in (3.25), the numerical consequent for each possible combination of the precedents in (3.22) can be calculated. However, the calculated value for each pair of precedent referential values must be transformed into a belief distribution to produce the belief degrees $\beta_{i,k}$ for each rule. Equation (3.11) can be used, with the calculated values using (3.25) for each rule, to produce a belief distribution of $\beta_{i,k}$ over the consequent values D using the substitutions

$$\alpha_{i,n} \to \beta_{i,k},\ \text{and}\ A_{i,} \to D. \tag{3.26}$$

This way all the belief degrees $\beta_{i,k}$ for each rule can be calculated in the initial rule base.

The remaining rule weights θ_k, and the attribute weights $\delta_{i,k}$, cannot be calculated using (3.25), and must be inferred in some other way. One way is to assume in the initial rule base for all the rules and attributes being of equal importance. In that case, the rule weights and attribute weights are set to be

$$\theta_k = \frac{1}{L} \text{ and } \delta_{i,k} = 1, \tag{3.27}$$

in the initial rule base according to (3.16) and (3.15).

All the necessary components of the initial rule base have now been defined. Input transformation for some observed attributes H can be done using (3.11). The defined packet precedent A can be used with H to produce the belief degrees $\alpha_{i,k}$, which can then be used in conjunction with the rule weights θ_k, and attribute weights $\delta_{i,k}$ to calculate the rule activation weights w_k using (3.14). Information of the Example 1's rule base can be summarized

Input	Belief output		
	D_1	D_2	D_3
$A_1(w_1)$	$\beta_{1,1}$	$\beta_{2,1}$	$\beta_{3,1}$
$A_2(w_2)$	$\beta_{1,2}$	$\beta_{2,2}$	$\beta_{3,2}$
$A_3(w_3)$	$\beta_{1,3}$	$\beta_{2,3}$	$\beta_{3,3}$
$A_4(w_4)$	$\beta_{1,4}$	$\beta_{2,4}$	$\beta_{3,4}$
$A_5(w_5)$	$\beta_{1,5}$	$\beta_{2,5}$	$\beta_{3,5}$
$A_6(w_6)$	$\beta_{1,6}$	$\beta_{2,6}$	$\beta_{3,6}$
$A_7(w_7)$	$\beta_{1,7}$	$\beta_{2,7}$	$\beta_{3,7}$
$A_8(w_8)$	$\beta_{1,8}$	$\beta_{2,8}$	$\beta_{3,8}$
$A_9(w_9)$	$\beta_{1,9}$	$\beta_{2,9}$	$\beta_{3,9}$

Table 2: *A belief rule expression matrix summarizing the information of the rule base for a rule base with nine belief rules, and three consequent referential values.*

in what is called a belief rule expression matrix shown in Table 2.

Each row in Table 2 represents one rule in the rule base with the associated consequent belief degrees. The first column in Table 2 represents the inputs, each being one of the pairs of referential values emerging from the Cartesian product in (3.22) to each rule with that rule's activation weight.

The information in Table 2 for some input X can now be used to infer the combined belief degrees of the BRB system using (3.20) to produce the output of the system S as defined in (3.17). Assuming a simple identity utility function with (3.18),

$$u(D_n) = D_n, \tag{3.28}$$

the numerical output of the BRB system can be finally calculated using (3.19) and the calculated combined belief degrees β_n. As an example of this, consider the belief distribution $\{0.25, 0.5, 0.25\}$ over the consequent referential values D in Example 1. The numerical

output according to the utility defined in (3.28) is

$$y = 0.25 \times D_1 + 0.5 \times D_2 + 0.25 \times D_3. \tag{3.29}$$

3.6 Training belief rule-based systems

By tuning the internal parameters of a BRB system: the rule weights θ_k, the precedent referential values A_i, the individual belief degrees $\beta_{i,k}$, and the attribute weights $\delta_{i,k}$, the BRB system can be trained to model some external system. By modelling an external system, it is meant that the BRB system is able to map inputs to outputs approximating the behavior of the system being modelled. This is done in the examples presented in Section 3.7. For example, a BRB model could be trained to model the behavior of a non-linear function of which the analytical form is not fully known, but the output to certain inputs are known.

One way of choosing suitable parameter values for a BRB system is to use adaptive training utilizing observed data of a system being modelled in the form of m input-output pairs $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$. Then, an initial BRB system can be used to compute an output $\mathbf{y}$ using a suitable utility function to transform the belief distribution over the consequent referential values into a form comparable to $\hat{\mathbf{y}}$ for every input $\hat{\mathbf{x}}$. The outputs produced by the BRB system can then finally be compared to the observed outputs, and using a suitable difference metric, the difference between the observed output and the output produced by the BRB system can be computed. By aggregating all of the differences, a cost function can be formulated. Assuming all the outputs to be scalar valued: $\hat{\mathbf{y}} = \hat{y}$ and $\mathbf{y} = y$, a cost function can be defined as

$$\xi(P) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2, \tag{3.30}$$

where the internal parameters of a BRB system are denoted as $P = P(\theta_k, A_i, \beta_{i,k}, \delta_{i,k})$, the chosen difference metric is the mean squared error, and the index i refers to the ith output

out of the m total outputs. Using the cost function (3.30), a minimization problem is defined

$$\min_{P \in \mathbf{P}} \xi(P), \tag{3.31}$$

where the parameters in P are subject to constraints indicated by $P \in \mathbf{P}$ in (3.31), and where $\mathbf{P}$ is the set of parameters P abiding the following constraints: the rule weights must sum to unity (3.24), the individual belief degrees for each consequent value in the kth rule must be less than or equal to one (3.7), and the precedent referential values are assumed to be hierarchical, that is, $A_{i,n} \leq A_{i,n+1}$ for all A_i present in the rule base. The constraint to keep the referential values hierarchical stems from the assumption that the input attributes x_i are monotonically increasing in terms of A_i, meaning, the higher the value of x_i is the higher the referential values $A_{i,n}$ to which x_i is assigned to are. Each of the parameters, apart from the precedent referential values, is also assumed to be constrained in the range $[0,1]$.

The problem in (3.31) can then be solved using conventional methods for solving differentiable single-objective optimization problems, like SLSQP as discussed in Section 2.3. The optimal solution $\tilde{P}$ contains the parameters for a BRB system that can reproduce outputs y close to the observed outputs $\hat{y}$, using the corresponding input vectors $\hat{\mathbf{x}}$. If the optimal parameters $\tilde{P}$ to the problem (3.31) results in the cost function value to be zero, then the resulting BRB system can predict the outputs corresponding to $\hat{\mathbf{x}}$ with a 100% confidence, which means that for all y, $y = \hat{y}$. Otherwise, a degree of error is present, which is directly proportional to the value of the cost function (3.31) with parameters $\tilde{P}$, and is defined as the mean squared error. After solving (3.31), the BRB system can be regarded as being trained to model an external system characterized by the observed input-output data.

3.7 A numerical example

In this section, a numerical example[4] is presented, where the ideas discussed in Section 3.5 and Section 3.6, are conceptualized into a practical example on how to build and train a simple BRB system.

[4]The code is available at `https://github.com/gialmisi/desdeo-brb/blob/master/desdeo_brb/example_nonlinear_scalar_function.py`

Suppose the non-linear scalar valued function

$$f : \mathbb{R} \to \mathbb{R} : f(x) = \cos(\sqrt{x})^2 + \cos(x)^2, \quad \text{where } x \in [0,5], \tag{3.32}$$

is to be modelled using a BRB system. The packet precedent of the BRB system consists of one vector of referential values, and is chosen by taking into account the limits imposed on the variable x in (3.32). The precedent referential values are chosen to be $A = A_1 = \{0,1,2,3,4,5\}$. The consequent referential values are defined to be $D = \{D_1, D_2, D_3, D_4, D_5\} = \{0, 0.5, 1, 1.5, 2\}$, where the first and last components are chosen based on the maximum an minimum values (3.32) can take, respectively. The rest of the components in D are chosen in a way to keep D evenly distributed, so that the referential values in the consequent can best represent any value between D_1 and D_5.

Using the function (3.32), the initial rules for the BRB system can be inferred. The value of (3.32) is computed for each of the precedent referential values in A. The resulting function value is then converted to a distribution of belief degrees using the substitutions described in (3.26), and (3.11). This results in six rules, $L = 6$, to be used in the BRB system.

The initial rule weights are chosen to be equal for each rule, that is $\theta_k = \frac{1}{L} = \frac{1}{6}$, $k \in [1,6]$, and the initial attribute weights are all chosen to be 1 in each rule: $\delta_{i,k} = \delta_{1,k} = \delta_k = 1$, $k \in [1,6]$. Each rule has now only one attribute weight, since the packet precedent A of the rule base, has only one element defined in it.

Rule k	θ_k	δ_k	x	$f(x)$	Consequent belief distribution $\{\ldots, (\beta_{i,k}, D_i), \ldots\}$
1	0.18	1	0	2.00	$\{(0,D_1),(0,D_2),(0,D_3),(0,D_4),(1.00,D_5)\}$
2	0.18	1	1	0.58	$\{(0,D_1),(0.83,D_2),(0.17,D_3),(0,D_4),(0,D_5)\}$
3	0.18	1	2	0.20	$\{(0.61,D_1),(0.40,D_2),(0,D_3),(0,D_4),(0,D_5)\}$
4	0.18	1	3	1.01	$\{(0,D_1),(0,D_2),(0.99,D_3),(0.01,D_4),(0,D_5)\}$
5	0.18	1	4	0.60	$\{(0,D_1),(0.80,D_2),(0.19,D_3),(0,D_4),(0,D_5)\}$
6	0.18	1	5	0.46	$\{(0,D_1),(0.08,D_2),(0.92,D_3),(0,D_4),(0,D_5)\}$

Table 3: *The initial rule base of the BRB system modelling the function* (3.32)*. The values tabulated are rounded to two decimals.*

Table 3 summarizes the parameters and the rules of the initial rule base defined. It is important to note that taking the sum of the consequent values, weighed by the individual belief degrees $\beta_{i,k}$, in each rule, should result in the corresponding tabulated $f(x)$ value in Table 3.

To train the rule base, a thousand[5] input attributes $\hat{x}$ are generated evenly distributed on the domain of (3.32). For each generated $\hat{x}$, an output $\hat{y}$ is computed using (3.32). This results in a thousand input-output pairs $(\hat{x}, \hat{y})$ that will act as observed data, as discussed in Section 3.6, which will be used to train the BRB system.

For each of the observed input attribute $\hat{x}$, an output y can be calculated using the BRB system according to (3.19), and the utility function defined in (3.18). The outputs y can be compared to the observed outputs $\hat{y}$, and a cost function (3.30) is formulated. Solving the minimization problem (3.31), the set of optimal parameters $\tilde{P}$ is obtained, which results in a BRB system able to produce from each input $\hat{x}$ an output y, which is close to the observed output $\hat{y}$. The lower the value of the cost function in (3.31) is, the closer the outputs of the BRB system will be to be observed outputs. The trained BRB system is summarized in Table 4.

Rule k	θ_k	δ_k	x	$f(x)$	Consequent belief distribution $\{\ldots,(\beta_{i,k},D_i),\ldots\}$
1	0.11	1	0	2.00	$\{(0.04,D_1),(0,D_2),(0,D_3),(0,D_4),(0.96,D_5)\}$
2	0.09	1	0.57	1.24	$\{(0.40,D_1),(0,D_2),(0,D_3),(0,D_4),(0.60,D_5)\}$
3	0.16	1	1.75	0.09	$\{(0.95,D_1),(0,D_2),(0,D_3),(0,D_4),(0.06,D_5)\}$
4	0.22	1	3.21	1.04	$\{(0.46,D_1),(0,D_2),(0,D_3),(0.04,D_4),(0.50,D_5)\}$
5	0.20	1	4.67	0.31	$\{(0.71,D_1),(0.18,D_2),(0,D_3),(0,D_4),(0.11,D_5)\}$
6	0.23	1	5	0.46	$\{(0.36,D_1),(0.56,D_2),(0,D_3),(0,D_4),(0.08,D_5)\}$

Table 4: *The trained rule base of the BRB system for modelling the function* (3.32)*. Tabulated values are rounded to two decimals.*

Computing the numerical output for each rule in Table 4 by taking the sum (3.19) over the consequents weighted by the belief degrees, and using the utility (3.28), will not necessarily exactly agree with the function's true output on the $f(x)$ column for the same input attribute

[5]Which is arguably a few hundred too many for such a simple problem.

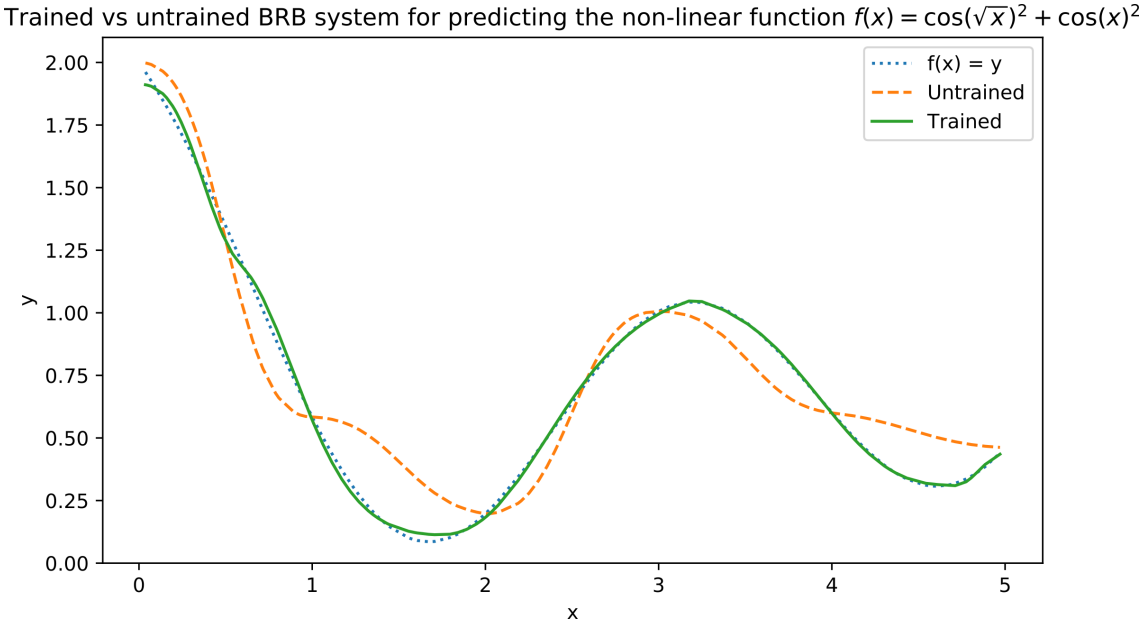


Figure 3: *The trained and untrained output of the BRB system for the evaluation attributes $\tilde{x}$ plotted alongside the true values of the function the BRB system is trying to model.*

x. This is the expected behavior, which results from the way the BRB parameters P are trained using the minimization problem (3.31), where the value of the final minimized cost function is greater than zero. The minimized value of the cost function resulting from the example in this section was 0.00045, therefore some margin of error is expected.

To test the trained BRB system, and compare it to its untrained initial state, where the BRB system's parameters P have not been computed by minimizing (3.31), a set of a hundred evaluation input attributes $\tilde{x}$ is generated, all differing from the observed input attributes $\hat{x}$. This way the predictive capability of the BRB system can be tested on data the system has not seen before.

In Figure 3, the output of both the trained and untrained BRB systems is plotted for the evaluation input attributes $\tilde{x}$, alongside with the actual function values $f(\tilde{x})$ computed using (3.32). It is evident from Figure 3 that the trained BRB system is more capable of modelling the original function. However, it is also worth noticing that the untrained BRB system is not far off from the original function either, which shows the predictive power of BRB systems with only the initial rule base being defined.

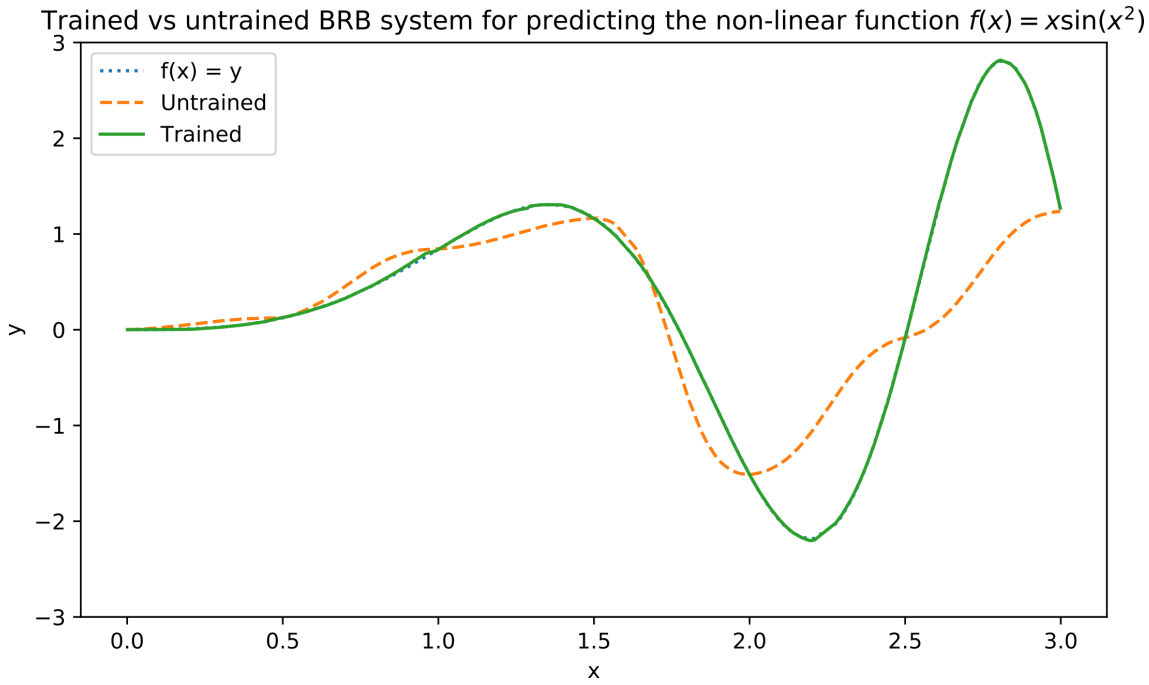


Figure 4: *The trained and untrained output of the BRB system modelling the function* $x\sin(x)^2$*, which was used in an example presented in [36].*

The example given in this section was calculated using the BRB software framework to be discussed in Chapter 4. Additionally, the original example in [36] has been reproduced using the same software framework.[6] In the original example, the function $x\sin(x^2)$ was modelled, but was otherwise identical to the numerical example in this section. The original example was reproduced to verify that the software framework works as intended. The results of the reproduction are depicted in Figure 4.

3.8 Examples of real-world applications

Belief-rule based systems have been used in real world applications as well. For example in [37], a BRB system has been used for the fault diagnosis of marine diesel engines. The used BRB system consists of many concurrently activated BRB systems, referred to as subsystems. Each subsystem is trained to predict a different kind of possible diesel engine fault, and a single input is fed concurrently to all the subsystems resulting in a concurrent BRB

[6]The code used to compute the results is available at `https://github.com/gialmisi/desdeo-brb/blob/master/desdeo_brb/example_original_article.py`

system. The authors of [37] argue that the concurrent BRB system is able to detect coexisting fault modes in marine diesel engines. The performance of the concurrent BRB system is also compared to more conventional machine learning models: artificial neural networks, support vector machines, and binary logistic regression. The authors conclude that the concurrent BRB system outperforms all the models it is compared to, both in accuracy of the fault predictions and stability. The concurrent BRB system is therefore concluded to be suitable for predicting faults in marine diesel engines.

Similarly in [38], BRB systems have been used to build a decision support system for physicians to predict in-hospital death and intensive care unit admission among trauma patients. The BRB based system is also compared to artificial neural networks, support vector machines, several logistic regression models, all using the same data set. The authors argue that the BRB system has the best predictive capabilities, and is possible to be improved further using historical data.

Additionally, in [39] a BRB system has been used for pipeline leak detection, and in [40] for portfolio optimization with non-linear cash flows and constraints. BRB systems have also been used to assess the clinical risk of cardiac chest pains under uncertainty in [41].

It is evident that BRB systems are a viable method to model a wide array of systems. They seem to be comparable, and arguably even better in some cases, when compared to conventional machine learning methods, such as neural networks and support vector machines.

4 The INFRINGER method

In this chapter the interactive multi-objective optimization method developed in this thesis is introduced. The method aims to combine BRB systems discussed in Chapter 3 with multi-objective optimization discussed in Chapter 2. The method is named INFRINGER, which stands for ***I****nteractive i****n****ference o****f*** *prefe****r****ences us****ing*** *beli****ef*** *****r****ules*. The name tries to capture the central idea behind the method.

This chapter will begin with Section 4.1 where the INFRINGER method and its two variants are described. Section 4.2 will give a brief description of the software framework developed by the author of this thesis to build and train belief-rule based systems. And finally, in Section 4.3 the software implementation of the INFRINGER method is discussed.

4.1 INFRINGER

INFRINGER is an interactive multi-objective optimization method, which aims to learn the preferences of a decision maker utilizing a value function discussed in Section 2.4.2, and by modeling the value function using a BRB system.

The central idea of the method is to interactively show a decision maker Pareto optimal objective vectors related to a multi-objective optimization problem being solved and interactively ask the decision maker for his/her preferences. Using the preference information, the underlying BRB system is trained to model the decision maker's preferences. The type of preference asked from the decision maker depends on which variant of the method is used. Two variants of the INFRINGER are described in this section: Variant 1 and Variant 2. In both variants, it is assumed that the preferences of the decision maker do not change during the course of the algorithm.

In what follows, the different steps of the INFRINGER method are described in detail. Based on the descriptions given, the method is then summarized in Algorithm 1.

4.1.1 Input and output

Common to both variants of INFRINGER, the method requires as its input the following:

1. A set of non-dominated objective vectors Z^{Pareto} approximating the Pareto optimal set of a multi-objective optimization problem.
2. The ideal point $\mathbf{z}^*$ and (representation of) the nadir point $\mathbf{z}^{\text{nad}}$ of the representation of the Pareto optimal set.
3. A fitness threshold $\gamma_{\text{th}} \in \mathbb{R}^+$ used to asses the fitness of the learned value function – How well does the learned value function approximate the true value function of the decision maker? – and to act as a termination criterion in the method.

The representation of the Pareto optimal set, the nadir point, and the ideal point, can be computed for a multi-objective optimization problem using the methods discussed in Section 2.3. The threshold can be set by either the analyst[1] or it can be decided between the decision maker and the analyst.

The output for both variants include:

1. A trained BRB system able to model the decision maker's preferences using a learned value function.
2. The five most preferred objective vectors in Z^{Pareto} according to the learned value function.

4.1.2 Conceptualizing the representation of the Pareto optimal set to the decision maker

After the required input to the INFRINGER method has been acquired, the nadir and ideal points are shown to the decision maker. He/she is then asked whether they would like to give a reference point $\bar{\mathbf{z}}$, with each element being restricted to the bounds spanned by the elements in the nadir and the ideal points. If the decision maker does not wish to give a

[1] Person in charge of formulating the problem and operating the method.

reference point, the reference point is set to be

$$\bar{\mathbf{z}} = \left\{ \frac{z_i^{\text{nad}} + z_i^*}{2} \mid i \in [1, m] \right\}, \tag{4.1}$$

where maximization is assumed for all objectives in the underlying optimization problem, and m is the number of objectives present in the optimization problem. The reference point acts as the mid-point when building the initial rule base of the underlying BRB system used in the method.

Next, Z^{Pareto}, $\mathbf{z}^{\text{nad}}$, $\mathbf{z}^*$, and $\bar{\mathbf{z}}$ are visualized to the decision maker. In this thesis, only multi-objective optimization problems with a maximum of three objectives are considered. It is therefore out of the scope of this thesis to go into details about techniques for visualizing $m > 3$ dimensional objective vectors. For discussions on how to visualize problems with $m > 3$ objectives, see [42] and [43] for example. Henceforth, a maximum of three objectives is assumed, and visualization techniques like 3-dimensional scatter plots can be assumed to be suitable. However, it is important to note that the INFRINGER method is not limited by the number of objectives present in the underlying multi-objective optimization problem, and could be applied to problems with more than three objective functions.

4.1.3 Belief rule-based system initialization

Using the reference point given, as discussed in Section 4.1.2, the BRB system of the INFRINGER method is initialized. The BRB system is built like described in Sections 3.5 and 3.7.

Internal data[2] used in the INFRINGER method is always assumed to be normalized. For the objective values, normalization means that all the objective values will have a maximum possible values of 1, and a minimum possible value of 0. The ideal and nadir points are used to compute the normalization, and are also normalized to the same scale. This means that internally, and assuming maximization, the nadir point will have all its components set to 0, and the ideal point will have all its components set to 1. The Pareto optimal objective vectors

[2]Data not shown to the decision maker, but used for internal computation in the INFRINGER method.

will then have all their components restricted between 0 and 1.

The packet precedent A of the belief rule-base is defined to have m finite referential value arrays A_i as its elements – one array for each objective. Each A_i has three precedent referential values as their elements, and each A_i defines the possible referential values for each of the objectives. The first element in each A_i is set to be 0, and the last element is set to be 1. The middle element is set to be $A_{i,2} = \bar{z}_i, i \in [1,m]$. The choice of three elements for each A_i has been a result of trial and error. However, the author of this thesis has concluded that three referential values for each objective offer a reasonable trade-off between predictive power and computational efficiency in the resulting BRB system. Consequent referential values are set to be $D = \{0, 0.25, 0.5, 0.75, 1\}$, the choice of which is also a result of trial and error. Because the choice of referential values, both for the precedents and consequent, are a result of trial and error, the formulation of the referential values could be experimented with for potentially enhancing the performance of INFRINGER further.

The rule base of the BRB system will have 3^m rules according to (3.24). The numerical output of each rule is computed using the principle

$$f_{\mathrm{p}}(\mathbf{z}) = \frac{\sum_{i=1}^{m} z_i}{m}, \tag{4.2}$$

which is chosen to represent a simplest possible initial value function congruent with the assumptions made in Section 2.4.2. In (4.2), the p present in the sub-index is part of the name of the function, and to remind the reader, $\mathbf{z}$ is an objective vector with components z_i. The computed value using (4.2) is then transformed in a belief distribution over the consequent referential values D, like discussed in Section 3.5. This kind of belief distribution is computed for each belief rule in the initial rule base with each rule being defined as presented in (3.6). The rest of the parameters in the BRB system: the rule weights and the attribute weights are initialized according to (3.27).

A simple identity utility function is chosen equivalent to (3.28). The BRB system can then be used to compute a numerical output for any of the objective vectors present in Z^{Pareto} as described in Section 3.4. The numerical value computed by the BRB system for an objective

vector $\mathbf{z}$ will be referred to as the score given by the BRB system for that particular vector. As a short-hand notation, let

$$\mathfrak{B}(\mathbf{z}) = \text{score calculated for } \mathbf{z} \tag{4.3}$$

denote the score computed by the BRB system for the objective vector $\mathbf{z}$. The score calculated using (4.3) will be limited between 0 and 1 as a direct consequence of the normalization of the parameters used in the BRB system.

According to (4.2), the nadir and ideal points will yield $\mathfrak{B}(\mathbf{z}^{\text{nad}}) = 0$ and $\mathfrak{B}(\mathbf{z}^{*}) = 1$, and (4.3) is also congruent to the assumptions made regarding the value function in Section 2.4.2. Therefore, the initial BRB system is assumed to model – or at least approximate closely – a valid value function.

4.1.4 Assessing the initial fitness of the model

With the BRB system initialized, a score according to the BRB system can be calculated for each objective vector in Z^{Pareto} using (4.3). The scores will form a set

$$S_{\text{BRB}} = \left\{ \mathfrak{B}(\mathbf{z}) \mid \mathbf{z} \in Z^{\text{Pareto}} \right\}. \tag{4.4}$$

Calculating the mean μ and standard deviation σ for the scores in (4.4), five objective vector pairs are chosen from Z^{Pareto} based on the values of μ and σ in the following way: the vectors are chosen such that the first pair consists of a pair of vectors with score values being a standard deviation σ apart from each other [3] centered around the mean μ. The second pair is picked similarly to the first pair, but 2σ apart, the third pair being 3σ apart. The fourth pair will consist of the objective vectors with the highest and lowest scores in the representation of the Pareto optimal set according to (4.4), and the fifth pair will consist of two randomly chosen objective vectors in the representation of the Pareto optimal set. The goal of choosing the pairs in the manner described is to get a representation of the objective vectors in the Pareto set with enough variety for the vectors to be interesting to the decision

[3] Or as close as possible to being one standard deviation apart in respect to their scores. The same is true for the rest of the pairs, respectively.

maker, and to capture preference information based on noticeably different objective vectors. The pairs are then shown to the decision maker.

For each pair of objective vectors, the decision maker is asked whether they prefer the first or the second vector or if the decision maker deems the vectors to be equally preferable. An objective vector being preferred to another means that its value should be higher than the other's according to the value function of the decision maker. Assuming a true value function s_{DM} exists, which is able to model the decision maker's true preferences, the preferences can be expressed using s_{DM} as

$$\begin{aligned} \mathbf{z}_1 \text{ is preferred to } \mathbf{z}_2: &\quad s_{\text{DM}}(\mathbf{z}_1) > s_{\text{DM}}(\mathbf{z}_2) \\ \mathbf{z}_2 \text{ is preferred to } \mathbf{z}_1: &\quad s_{\text{DM}}(\mathbf{z}_1) < s_{\text{DM}}(\mathbf{z}_2) \\ \mathbf{z}_1 \text{ is equally preferred to } \mathbf{z}_2: &\quad s_{\text{DM}}(\mathbf{z}_1) = s_{\text{DM}}(\mathbf{z}_2), \end{aligned} \tag{4.5}$$

where $\mathbf{z}_1, \mathbf{z}_2 \in Z^{\text{Pareto}}$. The BRB system in the INFRINGER method tries to model the true value function of the decision maker s_{DM}.

The preference information on the left hand side of the statements in (4.5) is given explicitly by the decision maker for the five pairs of objective vectors shown. The same pairs can be used with the BRB system, and the preference according to the value function modelled by the BRB system can then be compared to the preferences which were given by the decision maker. The true form of s_{DM} is never assumed to be known.

A metric indicating how well the value function modelled by the BRB system agrees with the true value function of the decision maker can then be computed as

$$f_{\text{fit}}(\mathbf{z}_1, \mathbf{z}_2) = \begin{cases} 1 \quad \text{if any of} \begin{cases} s_{\text{DM}}(\mathbf{z}_1) > s_{\text{DM}}(\mathbf{z}_2) \quad \wedge \quad \mathfrak{B}(\mathbf{z}_1) > \mathfrak{B}(\mathbf{z}_2) \\ s_{\text{DM}}(\mathbf{z}_1) < s_{\text{DM}}(\mathbf{z}_2) \quad \wedge \quad \mathfrak{B}(\mathbf{z}_1) < \mathfrak{B}(\mathbf{z}_2) \\ s_{\text{DM}}(\mathbf{z}_1) = s_{\text{DM}}(\mathbf{z}_2) \quad \wedge \quad \mathfrak{B}(\mathbf{z}_1) = \mathfrak{B}(\mathbf{z}_1) \end{cases}, \\ 0 \quad \text{otherwise}, \end{cases} \tag{4.6}$$

where the left hand sides of the conditionals for f_{fit} being 1, are replaced by the information

gathered from the decision maker according to (4.5). In other words, it is not necessary to know the form of s_{DM}.

Finally, a fitness γ can be computed

$$\gamma = \sum_{\text{for all five pairs } (\mathbf{z}_1, \mathbf{z}_2)} f_{\text{fit}}(\mathbf{z}_1, \mathbf{z}_2)/5 \tag{4.7}$$

to indicate a degree of agreement between the utility function $\mathfrak{B}$ being modelled, and the true value function for the decision maker s_{DM}. The fitness γ will take values between 0 and 1: 0 indicating a full disagreement and 1 indicating a full agreement. The fitness parameter γ can also be transformed into a percentage value multiplying it by 100. A percentage value can sometimes be more readily understood by humans than a value between 0 and 1.

4.1.5 The cost function

Before the two variants of the INFRINGER method are described, it is necessary to define the cost function used to optimize the internal parameters $P = P(\theta_k, A_i, \beta_{i,k}, \delta_{i,k})$ of the BRB system in a similar fashion to what was done in Section 3.6.

The cost function is defined as an aggregated cost function

$$\xi(P \to \mathfrak{B}) = (\xi_{\text{nadir}} + \xi_{\text{ideal}} + \xi_{\text{mono}} + \xi_{\text{scores}} + \xi_{\text{pair}})/5, \tag{4.8}$$

where $P \to \mathfrak{B}$ indicates that the underlying BRB system is always updated with the parameters P before it is evaluated, and the ξ's on the right hand side are other cost functions related to different criteria, which will be discussed in the remainder of this subsection.

In (4.8), ξ_{nadir} is the nadir cost function and ξ_{ideal} is the ideal cost function. The nadir and ideal cost function are defined as

$$\xi_{\text{nadir}} = \xi_{\text{nadir}}(P \to \mathfrak{B}; \mathbf{z}^{\text{nad}}) = \begin{cases} 0 & \text{if } |\mathfrak{B}(\mathbf{z}^{\text{nad}})| - \delta \leq 0, \\ |\mathfrak{B}(\mathbf{z}^{\text{nad}})| & \text{elsewhere,} \end{cases} \tag{4.9}$$

and

$$\xi_{\text{ideal}} = \xi_{\text{ideal}}(P \to \mathfrak{B}; \mathbf{z}^*) = \begin{cases} 0 & \text{if } |\mathfrak{B}(\mathbf{z}^*) - 1| - \delta \leq 0, \\ |\mathfrak{B}(\mathbf{z}^*) - 1| & \text{elsewhere,} \end{cases} \tag{4.10}$$

where in both equations δ is some small positive real value, which is used to allow the value of $\mathfrak{B}$ to differ from a target value by δ. The cost functions (4.9) and (4.10) are used to limit the value of $\mathfrak{B}$ to be between 0 and 1, such that it will score the nadir point to be as close as possible to 0 and the ideal point to be as close as possible to 1. If the output of $\mathfrak{B}$ is not in the target range, which depends on the magnitude of δ, then the cost function's value will be a positive value greater than zero.

The function ξ_{mono} in (4.8) is the monotonic cost function, and is defined as

$$\xi_{\text{mono}} = \xi_{\text{mono}}(P \to \mathfrak{B}; E_{\text{mono}}) = \sum_{\forall e_i \in E_{\text{mono}}} e_i, \tag{4.11}$$

where

$$E_{\text{mono}} = \left[\begin{cases} 0 & \text{if } \neg \bigvee_{i=1}^{m} (z'_i \geq z''_i) \wedge \mathfrak{B}(\mathbf{z}') \geq \mathfrak{B}(\mathbf{z}''), \\ \frac{1}{|E_{\text{mono}}|} & \text{else,} \end{cases} \middle| \, \mathbf{z}' \in Z' \text{ and } \mathbf{z}'' \in Z'' \right], \tag{4.12}$$

is a set of real-valued numbers, where each element e_i represents the monotonicity of (4.3) when tested with two vectors $\mathbf{z}'$ and $\mathbf{z}''$ according to a similar condition for monotonicity defined in (2.20), but with maximization assumed; if (4.3) is monotonic, e_i is zero, otherwise it will take on the value of the reciprocal of $|E_{\text{mono}}|$. In (4.12), Z' and Z'' are two sets of equal size, $|Z'| = |Z''|$, which consist of objective vectors with components limited to the range set by the nadir and ideal points defined in the underlying multi-objective optimization problem. The two sets have been chosen to contain randomly chosen objective vectors uniformly spread in the space limited by the ideal and nadir points. The elements in Z' and Z'' should be mutually unique, that is, no same objective vector should be present in both sets to avoid degeneracy. The unary operator "$\neg$" in (4.12) is the logical negation operator.

If all the elements e_i in E_{mono} are zero, it means that $\mathfrak{B}$ is monotonic at least in the discrete space spanned by Z' and Z'' and (4.11) will be equal to 0. Otherwise (4.11) will have a positive value greater than zero.

It is assumed in the INFRINGER method that if $\mathfrak{B}$ is monotonic in the discrete space spanned by Z' and Z'', then it is sufficiently monotonic elsewhere. This is an approximation. How good the approximation for the monotonicity of $\mathfrak{B}$ is depends on the quality of the spread of the objective vectors in both Z' and Z'' and the size of the sets. If Z' and Z'' are large sets with evenly spread vectors, then the approximation is assumed to be sufficiently good.

The cost function ξ_{score} in (4.8) represents the score cost function related to how well $\mathfrak{B}$ scores individual objective vectors compared to the scores the decision maker has assigned to the same vectors. This cost function ξ_{score} is only relevant in the first variant of the INFRINGER method. How the decision maker assigns these scores, is discussed in Section 4.1.6. The cost function ξ_{score} is defined as

$$\xi_{\text{score}}(P \to \mathfrak{B}; Z^s, S^s) = \frac{1}{n_s} \sum_{i=1}^{n_s} f_s(\mathbf{z}_i^s, s_i),\ \mathbf{z}_i^s \in Z^s, s_i \in S^s, \tag{4.13}$$

where

$$f_s(\mathbf{z}, s) = \begin{cases} 0 & \text{if } |\mathfrak{B}(\mathbf{z}) - s| - \delta \leq 0, \\ |\mathfrak{B}(\mathbf{z}) - s| & \text{elsewhere.} \end{cases} \tag{4.14}$$

In (4.13), Z^s and S^s are sets, where the index s refers to the sth objective vector in Z^s scored by the decision maker with score S^s, and n_s dentoes the number of objective vectors scored by the decision maker. Therefore, the set Z^s is composed of objective vectors, and S^s is a set of real valued numbers representing the scores given by the decision maker for the vectors in Z^s. Both sets are also of equal size, $|Z^s| = |S^s| = n_s$. The threshold δ in (4.14) plays a similar role to the δs present in the nadir and ideal cost functions in (4.9) and (4.10).

Lastly, the function ξ_{pair} denotes the cost function related to the pair-wise comparison of

objective vectors discussed in Section 4.1.4. It is defined as

$$\xi_{\text{pair}}(P \to \mathfrak{B}; \mathbf{Z}^{\text{pairs}}) = \sum_{\forall(\mathbf{z}_1,\mathbf{z}_2)\in\mathbf{Z}^{\text{pairs}}} f_{fit}(\mathbf{z}_1,\mathbf{z}_2)/n_p, \tag{4.15}$$

where $\mathbf{Z}^{\text{pairs}}$ is a set of objective vector pairs compared by the decision maker and $|\mathbf{Z}^{\text{pairs}}| = n_p$ denotes the number of pairs. If the value function being modelled is congruent with the decision maker's value function, then (4.15) will equal to zero.

All the cost functions defined in (4.8) have a maximum possible value of 1 and a minimum value of 0. Therefore, the value of (4.8) is also bound between 0 and 1. When (4.8) is minimized as a function of P, a set of optimal parameters $\tilde{P}$ is found. After the BRB system is updated with the optimal values $\tilde{P}$ found, the BRB system is assumed to be trained to model the preferences of the decision maker. Then, $\mathfrak{B}$ represents the decision maker's value function. The minimization problem can be formulated as

$$\min_{P\in\mathbf{P}} \xi(P \to \mathfrak{B}), \tag{4.16}$$

where the parameters P are restricted to the feasible set $\mathbf{P}$, which is defined by the same constraints discussed in Section 3.6.

4.1.6 Variant 1: scoring of individual points

In the first variant of the INFRINGER method, the decision maker is asked to score five objective vectors in Z^{Pareto}. The amount of five vectors is chosen because it is low enough to not tire the decision maker too much cognitively, yet high enough to provide enough information for the BRB system to be able to learn. The objective vectors to be shown to the decision maker are chosen by computing the mean μ and standard deviation σ for the scores in (4.4). The objective vectors are chosen based on their scores (4.4). The first objective vector is chosen to have a score as close as possible to the mean. The second and third objective vectors are chosen such that their scores are as close as possible to $\mu - \sigma$ and $\mu + \sigma$, respectively. The fourth and fifth vectors are chosen based on their scores being as close as possible to $\mu - 2\sigma$ and $\mu + 2\sigma$, respectively. The decision maker is then asked to

score each of the objective vectors on a scale from 0 to 100, where 0 is the worst possible score, indicating a very strong dissatisfaction in the objective vector, and where 100 indicates a very strong satisfaction in objective vector. This scale is chosen because it is assumed that this kind of scale is a familiar way for scoring things for the decision maker, and the scale also offers the possibility to fine-tune the scores if for example two vectors being scored are both almost equally preferable to the decision maker. This sort of fine-tuning is not possible if the scale for the scores was chosen to be from 1 to 5, for instance.

Using the objective vectors shown and the scores given by the decision maker, the cost functions (4.8) can be formulated and ξ_{pair} is set to be zero. The scores given by the decision maker are scaled to be between 0 and 1. Now (4.16) can be solved, and the found optimal value for the parameters P is used to update the BRB system. The resulting value function $\mathfrak{B}$ modelled by the updated BRB system is then visualized to the decision maker, and the partial ranking of the representation of the Pareto optimal set Z^{Pareto} is also visualized to the decision maker. The partial ranking is based on the scores given by the modelled utility function $\mathfrak{B}$. Examples of both visualizations mentioned can be found in Chapter 5.

Next, the fitness γ is calculated according to (4.7) by asking the decision maker to compare five pairs of objective vectors as was discussed in Section 4.1.4. If the fitness is satisfactory, meaning that it is greater than the fitness threshold γ_{th}, then the INFRINGER method terminates. If the fitness is not satisfactory, the cost function related to the pair-wise comparisons ξ_{pair} defined in (4.15) is formulated using the preference information given by the decision maker in the pair-wise comparisons. The cost function (4.8) is then updated and it is minimized again.

The same visualizations are presented to the decision maker, which were presented after the individual scoring of objective vectors and the fitness of the BRB system is once again calculated by asking the decision maker to compare five objective vector pairs. If the fitness is still not satisfactory, the new pair-wise comparisons are used to update in an additive way the cost function ξ_{pair}. Here, additive means that the information from the previous pair-wise comparisons is kept, with an exception made in regard the first pair-wise comparisons conducted from which comparison information is not kept as it is used to only assess the fitness of the initial value function modelled by the BRB system. Continuing, (4.16) is then

solved once again.

This whole process described, where the decision maker is asked to compare objective vector pairs, the fitness of the BRB system is calculated, and the cost function (4.16) is solved with an updated ξ_{pair} is repeated until a satisfactory fitness γ is reached. To quantify the termination criterion, the fitness threshold γ_{th} is used. When $\gamma > \gamma_{\text{th}}$ holds, the method is terminated.

4.1.7 Variant 2: pair-wise comparisons only

The second variant of the INFRINGER method is a simplified version of the first variant. Instead of individually scoring objective vectors, the preference information gathered from the initial pair-wise comparison is used to formulate ξ_{pair} from the very beginning. Then, (4.16) is solved immediately after the initial fitness of the BRB system is calculated and the method continues just as in the first variant. In the aggregated cost function (4.8), the cost function associated with the score ξ_{score} is set to be always zero in the second variant.

4.1.8 Termination

After a satisfactory fitness for the BRB system is reached, the INFRINGER method is terminated. Before the termination, the decision maker is shown the five highest scoring objective vectors in Z^{Pareto} according to $\mathfrak{B}$ from which the decision maker can then choose their most preferred solution. Five objective vectors are shown because the modelled value function will likely contain a margin of error dependant on the non-zero value of the aggregated cost function (4.8) when evaluated for the value function modelled by the final BRB system. If the value was zero, then it can be assumed that the highest scoring objective vector in the representation of the Pareto optimal set used in the INFRINGER method to be the best.

Supposedly, $\mathfrak{B}$ in the final BRB system is able to model the preferences of the decision maker in terms of his/her value function. The best objective vectors shown can be traced back to the corresponding decision variable vectors. However, these decision variable vectors can often be dimensionally large, as is the case in the case study discussed in Chapter 5 with near $1.3e4$ independent decision variables. It is not feasible to show the vectors in a numerical form,

but they should be visualized to the decision maker in one form or another. The visualization of such decision variable vectors is out of the scope of this thesis and will therefore not be discussed further.

4.1.9 Summarizing the INFRINGER method

The INFRINGER method described in this section is summarized in Algorithm 1 for both variants Variant 1 and Variant 2. Many of the abstract ideas, especially how the interaction with the decision maker takes place and the visualizations, are conceptualized in Chapter 5, where the INFRINGER method is used to solve a real-life problem in Finnish forestation.

4.2 The belief rule-based systems software framework

To build the necessary belief-rule based system used in the INFRINGER method, a software framework has been developed by the author of this thesis which has been implemented using Python [44]. In the software framework, the ideas discussed in Chapter 3 are implemented in an extensible and flexible modular software structure using object-oriented programming. The advantages that come with implementing the BRB system using object-oriented programming allow many of the individual components of a BRB system to be individually modified and accessed as needed. This is done for example in implementing the cost function specified in (4.8) by inheriting from a base cost function class and modifying the inherited version to be congruent with (4.8).

In the software framework implementation for building BRB systems, SciPy (the library)[4] [45], NumPy[5] [46], and Matplotlib[6] [47] software libraries have been used. All the software libraries used are part of the SciPy – which unfortunately shares the same name with the SciPy software library – Python-based ecosystem[7] of open-source software for scientific and numerical computing. Many of the equations used in the inference of a BRB system have been vectorized using the data structures available in NumPy, resulting in good com-

[4] `https://docs.scipy.org/doc/scipy/reference/` (July 3, 2020)
[5] `https://numpy.org/` (July 3, 2020)
[6] `https://matplotlib.org/` (July 3, 2020)
[7] `https://www.scipy.org/` (July 3, 2020)

putational efficiency. SciPy has been used to minimize the cost function with the sequential least squares programming method discussed in Section 2.3. Matplotlib has been used to implement visualizations.

The software framework can be used to define a BRB system with an arbitrary amount of input attributes. The only limitation is that the BRB system built should have a packed precedent A with elements A_i all having the same size. This limitation arises from the aforementioned vectorization, and has been a conscious trade-off made by the author of this thesis when developing said framework.

Section 3.7 contains a couple of examples on the usage of the framework for building BRB systems to model simple, nonlinear scalar valued functions with adaptive training. In Appendix A, a source code example to build and train a BRB system to model the Himmelblau[8] function is given. The example in Appendix A demonstrates how a BRB system with multiple input attributes can be modelled using the software framework. The software framework described in this section is available on GitHub[9] as freely distributable open-source software.

4.3 Implementation of the INFRINGER method

The INFRINGER method has been implemented in Python by the author of this thesis using the BRB software framework discussed in Section 4.2. The implementation has been used in the case study in Chapter 5, and the source code of the implementation is also available on GitHub[10]. The graphical user interface in the implementation of the INFRINGER method has been implemented using the bindings to the Tk graphical user interface toolkit[11] present in the Python library tkinter[12].

[8] $f(x,y) = (x^2+y-11)^2+(x+y^2-7)^2$

[9] `https://github.com/gialmisi/desdeo-brb` (July 3, 2020)

[10] `https://github.com/gialmisi/desdeo-brb/blob/master/desdeo_brb/method.py` (July 3, 2020)

[11] `https://www.tcl.tk/` (July 3, 2020)

[12] `https://docs.python.org/3/library/tkinter.html` (July 3, 2020)

Algorithm 1 The INFRINGER method

INPUT:

1. A set of non-dominated objective vectors approximating a Pareto optimal set: Z^{Pareto}.
2. The ideal and (approximation of the) nadir points, $\mathbf{z}^*$ and $\mathbf{z}^{\text{nad}}$, of Z^{Pareto}.
3. A fitness threshold γ_{th} to asses the learned model's fitness, and to act as a termination criterion.

OUTPUT:

1. A BRB system trained to model a decision maker's preferences using a value function.
2. The five most preferred objective vectors in Z^{Pareto}, according to the value function learned by the trained BRB system.

Step 1: Show the decision maker $\mathbf{z}^*$ and $\mathbf{z}^{\text{nad}}$, and ask if he/she would like to supply a reference point $\bar{\mathbf{z}}$ with each objective value being between the corresponding values in $\mathbf{z}^*$ and $\mathbf{z}^{\text{nad}}$.

Step 2: If $\bar{\mathbf{z}}$ was given in Step 1, continue to Step 3. Otherwise set $\bar{\mathbf{z}}$ to be an objective vector, with objective values being in the middle of the objective values in $\mathbf{z}^*$ and $\mathbf{z}^{\text{nad}}$.

Step 3: Visualize Z^{Pareto}, $\mathbf{z}^*$ and $\mathbf{z}^{\text{nad}}$, and $\bar{\mathbf{z}}$, to the decision maker.

Step 4: Initialize the BRB system as described in Section 4.1.3.

Step 5: Ask the decision maker to compare five pairs of objective vectors in Z^{Pareto} chosen according to Section 4.1.4.

Step 6: Assess the fitness γ of the initial BRB system according to (4.7) using preference information gathered in Step 5.

Step 7: If $\gamma \geq \gamma_{\text{th}}$, go to Step 10.

if Variant 1 **then**

Step 8.1: Ask the decision maker to score five objective vectors chosen from Z^{Pareto} as described in Section 4.1.6.

Step 8.2: Formulate the cost function (4.8) of the BRB system using preferences given in Step 8.1.

Step 8.3: Solve (4.16), and update the BRB system parameters with the optimal P found.

Step 8.4: Visualize to the decision maker his/her learned value function, and the ranking of the objective vectors in Z^{Pareto} according to the learned value function.

Step 8.5: Repeat Steps 5 and 6, and update γ. Continue to Step 8.6.

Step 8.6: If $\gamma \geq \gamma_{\text{th}}$, go to Step 10.

Step 8.7: Update (4.8) according to the preference information gathered in Step 8.5, and go to Step 8.3.

if Variant 2 **then**

Step 9.1: Formulate the cost function (4.8) of the BRB system using preferences given in Step 5.

Step 9.2: Solve (4.16), and update the BRB system parameters with the optimal P found.

Step 9.3: Visualize to the decision maker his/her learned value function, and the ranking of the objective vectors in Z^{Pareto} according to the learned value function.

Step 9.4: Repeat Steps 5 and 6, and update γ. Continue to Step 9.5.

Step 9.5: If $\gamma \geq \gamma_{\text{th}}$, go to Step 10.

Step 9.6: Update (4.8) according to the preference information gathered in Step 9.4, and go to Step 9.2.

Step 10: Show the decision maker the five most preferred objective vectors in Z^{Pareto} according to the learned value function in (4.3).

Step 11: End.

5 Case study

In this chapter, a case study is conducted using the INFRINGER method discussed in Chapter 4 with the aid of an expert decision maker. The purpose of the case study is to test the INFRINGER method with a real world multi-objective optimization problem, see how the method works in practice in aiding the decision maker, and how the method is able to learn the preferences of the decision maker. Additionally, in this chapter many of the topics left on an abstract level in Chapter 4 will be conceptualized.

5.1 A case study in Finnish forestation

5.1.1 Introduction and motivation

The case study discussed in this chapter is a problem in Finnish forestation, where different management strategies must be chosen for different areas of a forest to be executed during a certain time period. These areas are referred to as forest stands, which are the individual managerial parts a forest is divided into. Each strategy chosen for a forest stand has economical and environmental consequences. It is possible to simulate the effects of employing certain strategies for different forest stands in a forest using a simulation framework like discussed in [48]. However, the decision on what strategies to employ must still be made by a human decision maker.

The impact of the strategies chosen for each forest stand is simulated over a certain amount of years. Each strategy will cause the forest stand[1] to change. This change is measured by indicators like income produced by selling the harvested timber from a stand and the total carbon-dioxide stored in the stand. These indicators are the conflicting criteria the decision maker will base his/her decision on.

To aid the decision maker, different criteria must be quantified to him/her and the decision maker should be allowed to explore different options available. Assuming the decision maker to be rational like discussed in Section 2.4.1, it is assumed that he/she is only interested in

[1] Unless the stand is chosen to be set aside.

the objective vectors in the Pareto optimal set. Therefore, before the problem can be solved by the decision maker with the aid of the INFRINGER method, a representation of the Pareto optimal set must be generated.

5.1.2 Available data

Simulation data on Finnish forestation averaged over a one hundred year time horizon, is available for 1475 forest stands, each forest stand employing one possible strategy out of 59 available strategies. The data has been simulated using the simulation framework SIMO described in [48]. For each stand and strategy chosen, the following indicators for the consequences for employing a certain management strategy for a stand were chosen from the data:

1. Total average income.
2. Average stored carbon-dioxide.
3. Average combined habitat suitability index.

Other indicators were also available, but not used in the case study.

The data was available in multiple CSV[2] files, each file containing values of one indicator for each forest stand and for all available strategies. In each file, each row contained the indicator values for one stand separated in individual columns for each strategy. The first column and first row contained header information. The first row contained column names and the first column contained stand identification numbers. If some strategy was not applicable to a stand, the value on the column on that stand's row contained a blank value.

A total of 20 strategies were chosen to be considered in the case study together by the author of this thesis and the decision maker in the case study. Stands that were not possible to be managed by all of the 20 chosen strategies had the blank value replaced by a zero. Another option would have been to drop the incompatible stands from the final data completely, but because the small amount of initial data available, it was decided to not drop any stands from the data. In hindsight, this was not a good decision because indicators like stored carbon-

[2]Comma separated values.

dioxide would be set to be zero for a stand incompatible with a strategy. This would lead to a smaller value in the average value over all stands compared to what would result if the incompatible stands were just left aside, indicating no strategy being employed for the stand. However, the data was still applicable in the case study, and the aforementioned oversight regarding blank values did not introduce any unusual anomalies in the final computed representation of the Pareto optimal set, or at least the decision maker in the case study did not seem to notice any.

5.1.3 Modelling the problem

The problem in Finnish forestation can be modelled as a discrete data-based integer multi-objective optimization problem, with three objective functions to be maximized:

$$\begin{aligned} \max_{\mathbf{X}} \quad & \{f_1(\mathbf{X}), f_2(\mathbf{X}), f_3(\mathbf{X})\} \\ \text{s.t.} \quad & \sum_{i=1}^{n_{\text{strat}}} x_i = 1, \quad \text{where } x_i \text{ are components in } \mathbf{x}, \forall \mathbf{x} \in \mathbf{X}, \end{aligned} \tag{5.1}$$

where f_1 is the total average income summed over all stands, f_2 is the average stored carbon dioxide summed over all stands, and f_3 is average combined suitable habitat index summed over all stands, $\mathbf{x}$ is a decision variable vector with $n_{\text{strat}} = 20$ components x_i, and $\mathbf{X}$ is the set containing all the decision variable vectors $\mathbf{x}$ in the problem. The functions f_1, f_2 and f_3 are defined later in this subsection.

The jth component of $\mathbf{X}$, namely $\mathbf{x}_j$, indicates the management strategy employed for the jth forest stand. The components of $\mathbf{x}_j$ are all binary values, restricted to be either 0 or 1. The constraint in (5.1) implies that only one of the components in a decision variable vector $\mathbf{x}_j$ may have the value 1, while all the other values must be 0. In other words, only one management strategy can be chosen to be employed for each stand. Therefore, the ith component of $\mathbf{x}_j$ being 1, indicates that the ith available management strategy has been chosen for the stand that corresponds to $\mathbf{x}_j$. The set $\mathbf{X}$ consists of $n_{\text{stand}} = 1475$ elements.

Objective functions present in (5.1) can each be formulated as follows:

$$f_i(\mathbf{X}) = \sum_{j=1}^{n_{\text{stand}}} \mathbf{x}_j (\mathbf{s}_j^i)^T, \quad \mathbf{x}_j \in \mathbf{X}, \mathbf{s}_j^i \in \mathbf{S}^i, \tag{5.2}$$

where $\mathbf{S}^i$ is a set of n_{stand} real-valued vectors $\mathbf{s}^i$, where the jth element $\mathbf{s}_j^i$ corresponds to the values of the indicators of the jth stands for each available management strategy. This means that the kth component of the vector $\mathbf{s}^i$, namely s_k^i, is the indicator value when the kth available strategy is chosen for the stand corresponding to $\mathbf{s}^i$. The index i in $\mathbf{S}^i$ and $\mathbf{s}^i$, indicates which indicator the values in the vectors $\mathbf{s}_i$ correspond to. Total average income corresponds to $i = 1$, the average stored carbon corresponds to $i = 2$, and the average combined suitable habitat index corresponds to $i = 3$. In the summand of (5.2), the product between $\mathbf{x}_j$ and $(\mathbf{s}_j^i)^T$ is a matrix product, where the former vector is in row-form, and the latter in column-form indicated by the transposition $()^T$.

All the values contained in the elements of the sets $\mathbf{S}^1, \mathbf{S}^2$, and $\mathbf{S}^3$, are discrete. The values are read from a CSV file provided for each indicator, like was described in Section 5.1.2.

5.1.4 Computing the representation of the Pareto optimal set

To compute a representation of the Pareto optimal set for the problem in (5.1), the ideal and nadir points were calculated using a payoff table. Next, a set of evenly spaced objective value vectors in the objective space was generated akin to a grid, with individual objective values being limited by the ideal and nadir point; the maximum values for each objective limited by the ideal values, and the minimum values limited by the nadir values.

Each of the vectors generated was then used in a GUESS achievement scalarizing function, defined in (2.16), as a reference point. This resulted in many individual single-objective optimization problems, which were each solved using the Coin-or branch and cut method discussed in Section 2.3, because it is suitable for solving integer based problems. Each solved single-objective optimization problem resulted in one Pareto optimal objective vector. The values of the components of the vectors are plotted in Figure 5 as a 3-dimensional scatter plot, and in Figure 6 as three projections on the three different planes. Plots in both figures

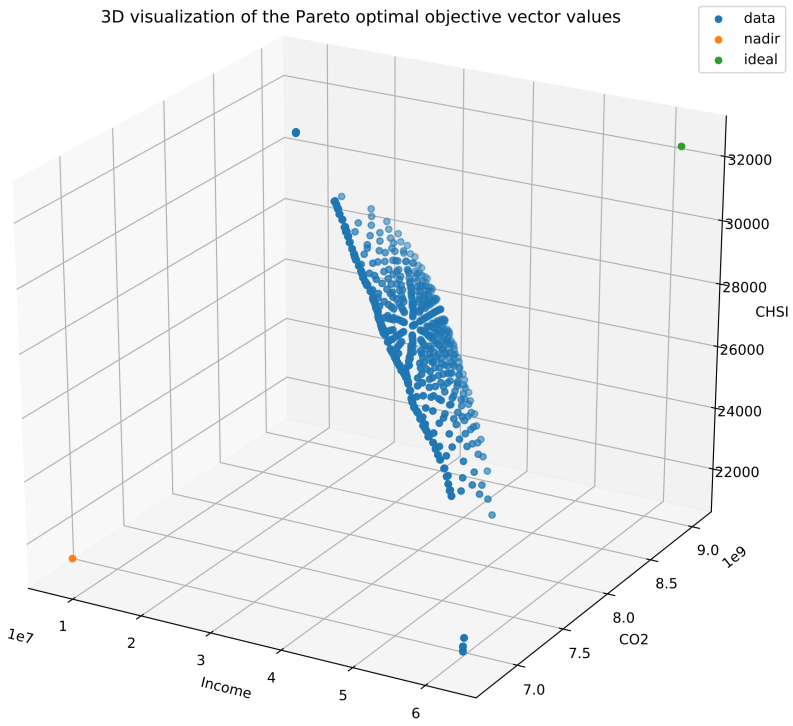


Figure 5: *The ideal, the nadir, and the Pareto optimal objective vectors computed for the problem in Finnish forestation, plotted as a 3-dimensional scatter plot.*

also include the computed ideal and nadir points. In the aforementioned figures, the average income is labeled as *income*, the average stored carbon-dioxide is labeled as *CO2*, and the average combined suitable habitat index is labeled as *CSHI*.

The set of Pareto optimal objective vectors computed using an achievement scalarizing function was additionally modified by removing all duplicate vectors. What was left, was a set of mutually non-dominated objective vectors, which were used as a representation of a Pareto optimal set for the Finnish forestation problem in this case study. The original number of reference point generated was around two thousand which resulted in around a thousand Pareto optimal objective vectors.

In practice, the problem was modelled in Python[44] using Pyomo [49, 50]: a Python software library implementing an optimization modelling language. The code[3] used by Jose

[3] `https://github.com/josejuhani/gradu-code` (July 3, 2020)

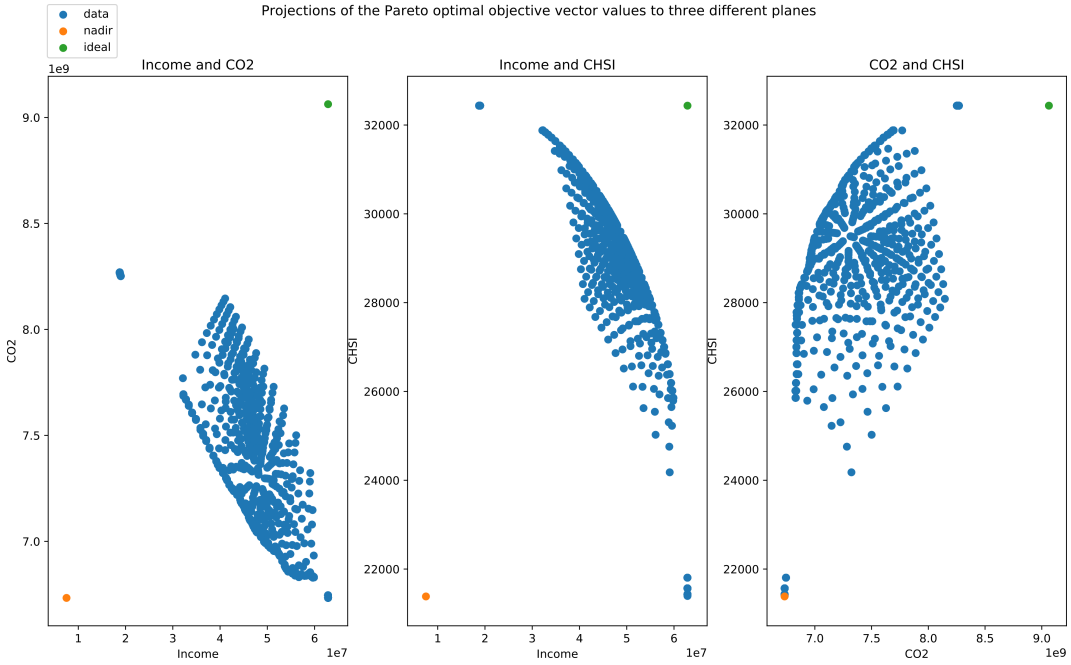


Figure 6: *The ideal point, the nadir point, and the values of the Pareto optimal objective vectors computed for the problem in Finnish forestation, plotted as 2-dimensional scatter plots in three different planes.*

Malmberg in his MSc thesis [51] was used in part to model the optimization problem in this thesis. Code used to generate the representation of the Pareto optimal set presented in this chapter is available on GitHub [4] alongside the original data used.

5.2 INFRINGER in practice

In this section, the representation of the Pareto optimal set computed for the problem described in Section 5.1 is used in the INFRINGER method, which was described in Chapter 4. With the aid of a human decision maker with expertise in Finnish forestation, the INFINGER method is used to aid the decision maker in a decision making process. The value function of the decision maker is also learned using the method. Both variants of the INFRINGER method were used. In what follows, *analyst* will refer to the author of this thesis. A fitness threshold value of $\gamma_{\text{th}} = 1$ was chosen in both variants of the method tested.

[4] `https://github.com/gialmisi/forest-opt` (July 3, 2020)

The session with the human decision maker documented in this section took place over a video call using Skype[5]. The INFINGER method was ran on the laptop of the analyst, who was also in charge of operating the method. The analyst shared the screen of his laptop over Skype with the decision maker. The whole session was recorded for both audio and video. The session had to be stopped soon after the test of the first variant of INFRINGER had begun due to a technical problem, where the analyst had to reboot his laptop. The recording resumed soon after, and was finished without further problems.

No particular research methods were employed in conducting the practical tests documented in this section. The results are mainly anecdotal and qualitative in nature and should be treated as such. That being said, the results of the practical test were still valuable in analyzing the effectiveness of the INFINGER method as an interactive multi-objective optimization tool, and in analyzing how well the method was able to learn the value function of the decision maker. The information gained in this case study can be used to further develop the INFRINGER method.

5.2.1 Before the tests

Before any of the tests were conducted, the decision maker was asked if he was willing to participate in the test and if the information gained during the tests could be used in this thesis. It was also asked if the decision maker would be comfortable with the whole test session being recorded. The decision maker's answers were all affirmative.

Next, the decision maker was briefly presented the different phases of the INFRINGER method, with the analyst giving dummy preference information when prompted for in the method. The aim was to make the decision maker familiar with the different steps of the INFRINGER method shown in Algorithm 1.

5.2.2 Testing the first variant

The decision maker was shown the nadir and ideal points of the representation of the Pareto optimal set like shown in Figure 7 and according to Step 1 in Algorithm 1. The deci-

[5] `https://www.skype.com` (July 3, 2020)

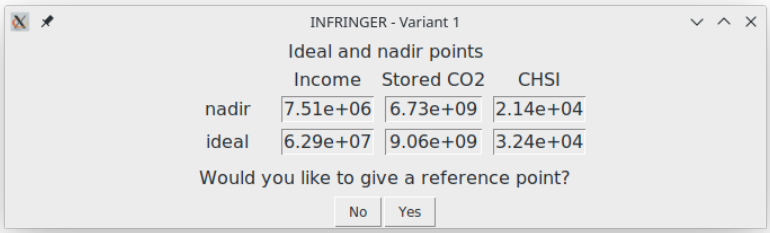


Figure 7: *The nadir and ideal points shown to the decision maker during the case study. The same points were used in both tests of the two variants of the INFRINGER method.*

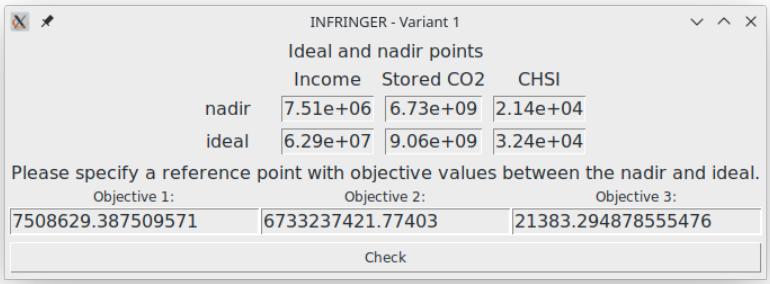


Figure 8: *Input of the reference point in the INFRINGER method.*

sion maker was then asked whether he would like to give a reference point according to Step 2 in Algorithm 1. After a while of thinking, the decision maker decided that he wanted to give a reference point. The reference point was given to the method in a window like shown in Figure 8. The actual reference point chosen by the decision maker was $\{2 \times 10^7, 8.2 \times 10^9, 2.8 \times 10^4\}$ for the income, CO2, and CSHI, respectively. The reference point was given by the decision maker in the original scale of the objectives and the units were omitted as they were not relevant during the test. The decision maker expected the trade-offs between the objectives to be inversely proportional to each other.

After the reference point was given according to Step 3 in Algorithm 1, the decision maker was shown the representation of the Pareto optimal set, the nadir point, the ideal point, and the given reference point in a three-dimensional scatter plot like shown in Figure 9. After

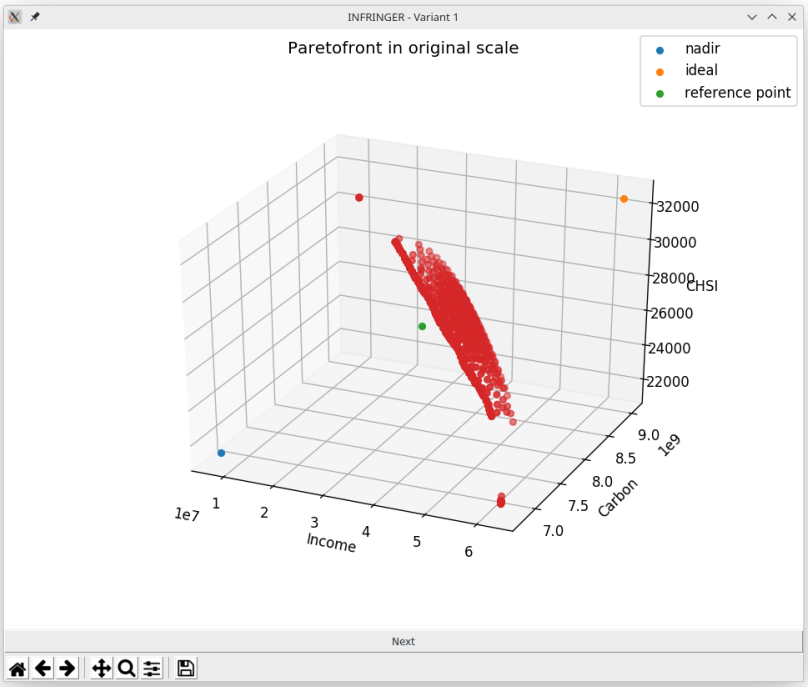


Figure 9: *An example of the initial representation of the Pareto optimal set shown to the decision maker after the reference point has been determined in the INFRINGER method. The nadir, the ideal, and the reference point are also shown. In the title on the plot, 'Paretofront' refers to the (representation of the) Pareto optimal set.*

seeing the representation of the Pareto optimal set, the decision maker concluded that his initial reference point given was less optimistic than he expected.

At this point, the laptop of the analyst froze and it had to be rebooted. The test was then continued and the BRB system was initialized as described in Step 4 in Algorithm 1.

Continuing, the decision maker was asked to give his preference by conducting a pair-wise comparison of five pairs of Pareto optimal objective vectors as described in Step 5 in Algorithm 1. The results of the comparisons can be found in Table 6. An example of the interface the decision maker was shown when asked to conduct the pair-wise comparisons can be seen in Figure 10 and an example of the radar plot shown to the decision maker for comparing

Candidate	Objective values			Score
	Income	CO2	CHSI	
1	5.69e7	7.15e9	2.52e4	30
2	5.59e7	6.84e9	2.78e4	30
3	4.79e7	7.13e9	2.97e4	50
4	5.11e7	7.70e9	2.66e4	70
5	4.56e7	7.73e9	2.90e4	80

Table 5: *The scores given by the decision maker to the Pareto optimal objective vectors shown while testing the first variant of the INFRINGER method.*

a single pair is shown in Figure 11. The initial fitness calculated was 60%[6] based on the pair-wise comparisons as described in Step 6 of Algorithm 1.

After the initial fitness was calculated and assessed according to the Step 7 in Algorithm 1, the INFRINGER method was continued. The decision maker was asked to score five Pareto optimal objective vectors as described in Step 8.1 in Algorithm 1. The decision maker asked at this point how the scores are related to each other, and the analyst explained that the scores are relative to each other. The decision maker seemed to understand what the analyst meant. To aid the decision maker in comparing the vectors, a parallel coordinate plot, of which an example can be seen in Figure 12, was also shown to him. The scores given by the decision maker can be seen in Table 5. Based on the scores given, the cost function of the INFRINGER method was formulated and optimized according to Step 8.2 and 8.3 in Algorithm 1.

The first time training the BRB model in the INFRINGER method took over two minutes, as is seen from Table 7. After the training, a plot of the learned value function was shown to the to the decision maker and a three-dimensional scatter plot of the ranked Pareto optimal objective vectors using the learned value function was shown right after according to Step 8.4 in Algorithm 1. Examples of both plots can be seen in Figures 13 and 14, respectively.

[6]Equivalent to $\gamma = 0.6$

Pair	Candidate 1 objective values			Candidate 2 objective values			Preference	Comments
	Income	CO2	CHSI	Income	CO2	CHSI		
Iteration 1 ($\gamma = 0.60$)								
1	4.19e7	7.27e9	3.08e4	4.87e7	7.02e9	2.96e4	=	Cannot see difference.
2	5.13e7	7.73e9	2.61e4	5.00e7	6.98e9	2.93e4	2	
3	5.11e7	7.70e9	2.66e4	5.59e7	6.84e9	2.78e4	1	
4	3.62e7	7.94e9	3.10e4	6.29e7	6.73e9	2.14e4	1	
5	6.29e7	6.73e9	2.14e4	3.62e7	7.94e9	3.10e4	2	Same as pair 4, but candidate 1 and 2 switched.
Iteration 2 ($\gamma = 0.40$)								
1	5.31e7	7.23e9	2.82e4	4.57e7	7.14e9	3.02e4	2	
2	4.24e7	7.35e9	3.07e4	4.62e7	7.11e9	3.01e4	=	
3	5.29e7	7.32e9	2.80e4	4.78e7	7.04e9	2.98e4	2	Not quite sure about preference.
4	3.62e7	7.94e9	3.10e4	6.29e7	6.73e9	2.14e4	1	
5	4.87e7	7.40e9	2.91e4	3.62e7	7.94e9	3.10e4	2	Second candidate same as first candidate in pair 4.
Iteration 3 ($\gamma = 0.60$)								
1	4.70e7	7.18e9	2.99e4	4.78e7	7.05e9	2.98e4	=	
2	4.12e7	7.30e9	3.09e4	5.19e7	7.11e9	2.88e4	1	
3	3.22e7	7.69e9	3.19e4	4.69e7	7.00e9	2.94e4	=	Decision maker felt candidates to be equally bad.
4	4.16e7	7.94e9	2.94e4	6.29e7	6.73e9	2.14e4	2	
5	4.16e7	7.94e9	2.94e4	6.29e7	6.73e9	2.14e4	2	Same as pair 4.

Table 6: *The preference information given by the decision maker in each iteration while testing the first variant of the INFRINGER method. The equal sign indicates the two candidates were equal preference wise. The fitness after each comparison is also reported for each iteration.*

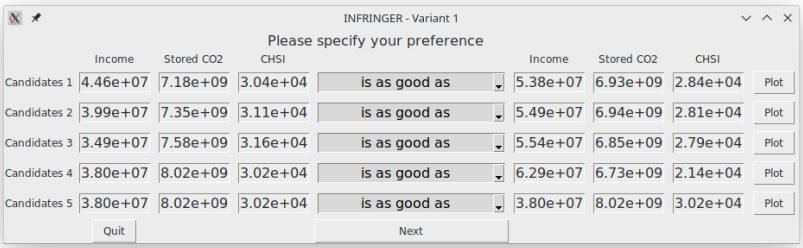


Figure 10: *Example of the interface shown to the decision maker in the INFRINGER method when asked to compare five pairs of Pareto optimal objective vectors, which are referred to as candidates in the figure. In the middle, the drop down menu offers three alternatives: 'is as good as', 'is better than', and 'is worse than', indicating which vector in each pair is preferred. On the right, the 'Plot' button may be pressed to show a radar plot of the two candidates, like shown Figure 11*

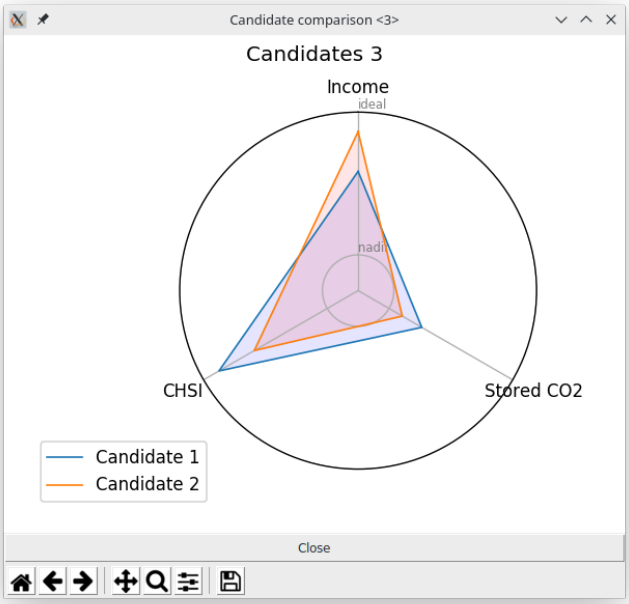


Figure 11: *Example of the radar plot shown to the decision maker in the INFRINGER method, during the pair-wise comparisons. Both objective vectors are shown, and are called candidates in the figure. The radar plot is scaled between the nadir and ideal points, where the ideal values are on the outer edge of the plot, and the nadir values are on the inner circle, for each of the three objectives.*

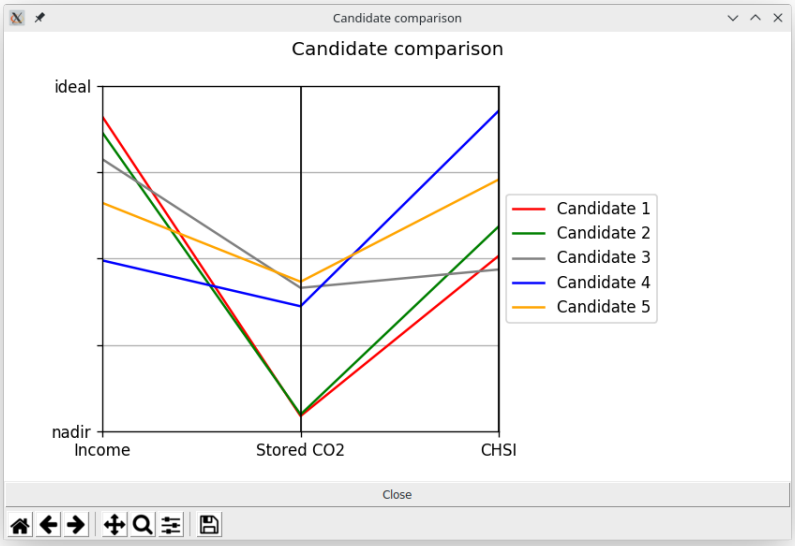


Figure 12: *An example of the parallel coordinates plot shown to the decision maker in the INFRINGER method to help him/her compare the five Pareto optimal objective vectors, referred to as candidates in the figure, in the first variant of the method. The nadir and ideal points are also shown in the figure.*

Iteration	Optimal cost function value	Training time
Variant 1		
1	0.0049	1min 59s
2	0.013	1min 33s
3	0.015*	7min 45s
Variant 2		
1	2.0e-18	40s
2	0.0015	1min 10s
3	0.0030	26s
4	0.0070	18s
5	2.2e-17	24s
6	0.0068	39s

Table 7: *The optimal value found in each iteration for the cost function during the testing of both variants of the INFRINGER method. The elapsed time in minimizing the cost function is also shown. The computations to minimize the cost functions were executed on a Dell Latitude 7480 laptop equipped with an Intel Core i5-7300U processor clocked at 2.60GHz per core. The star in the table indicates that the last iteration of the first variant had to be halted before an optimal value was found. The value trailing the star was the best value found before the termination.*

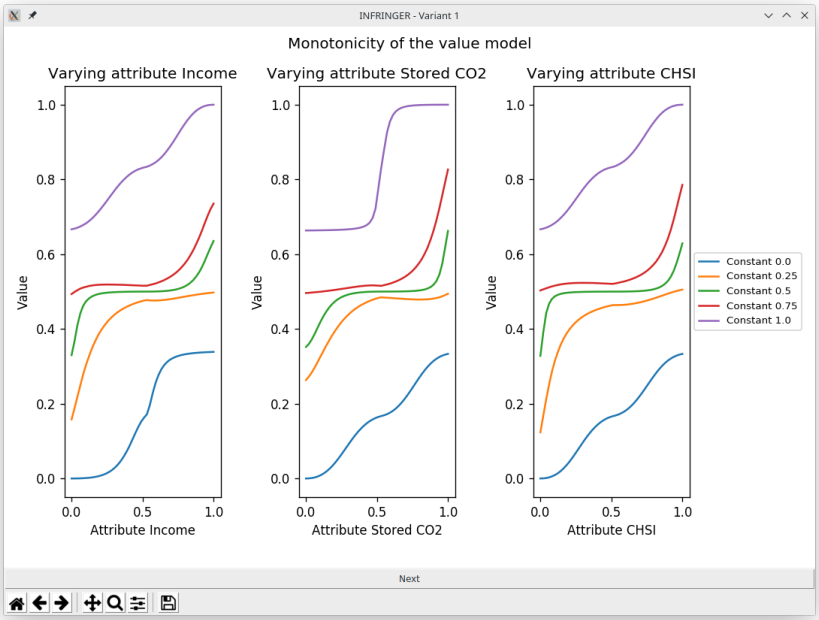


Figure 13: *An example of the value function shown to the decision maker in the INFRINGER method. In each of the three plots, one of the attributes, referring to an objective function's values, is varied, while the others are kept constant. Plots for five different constant values, for the objectives not being varied, are shown in each plot. The values are all scaled to be between 0 and 1, where 0 is the nadir point's value for each objective, and 1 is the ideal point's value for each objective. The value of the value function is shown on the vertical axis in each plot, and its value varies between 0 and 1.*

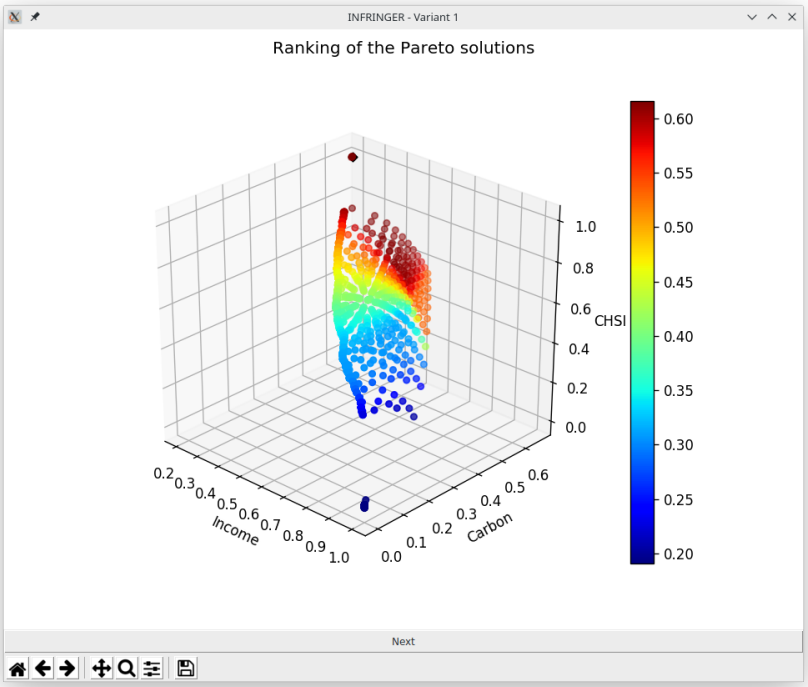
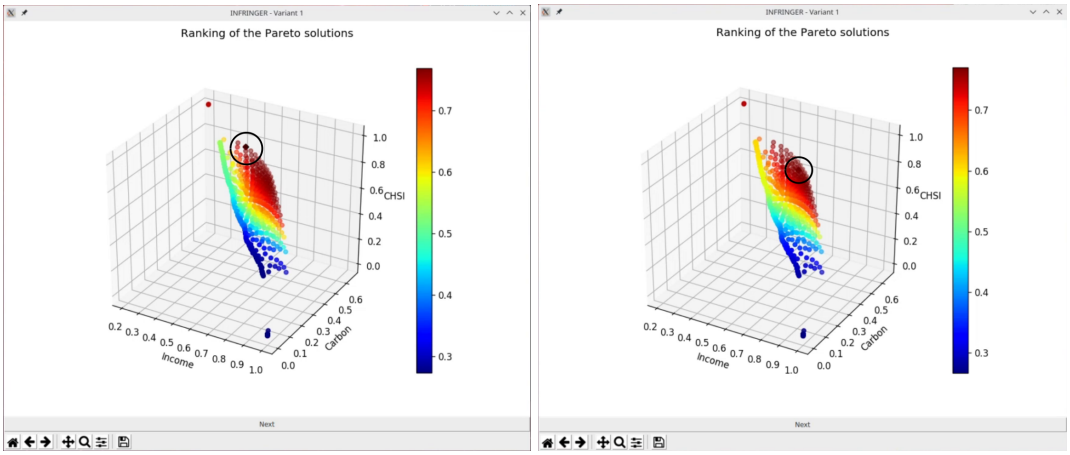


Figure 14: *An example of the ranked Pareto optimal objective vectors shown to the decision maker in the INFINGER method. The value computed using the learned value function is shown as a color for each Pareto optimal objective vector. The maximum possible value is 1 represented by a dark red color, and the minimum is 0 represented by a deep blue color. A black diamond is also shown (at the very top of the plot) indicating the point having the highest value according to the value function. Each of the axes is scaled between 0 and 1, according to the nadir and ideal points, similarly to Figure 13.*



(a) *Ranking of the objective vectors in iteration 1.*

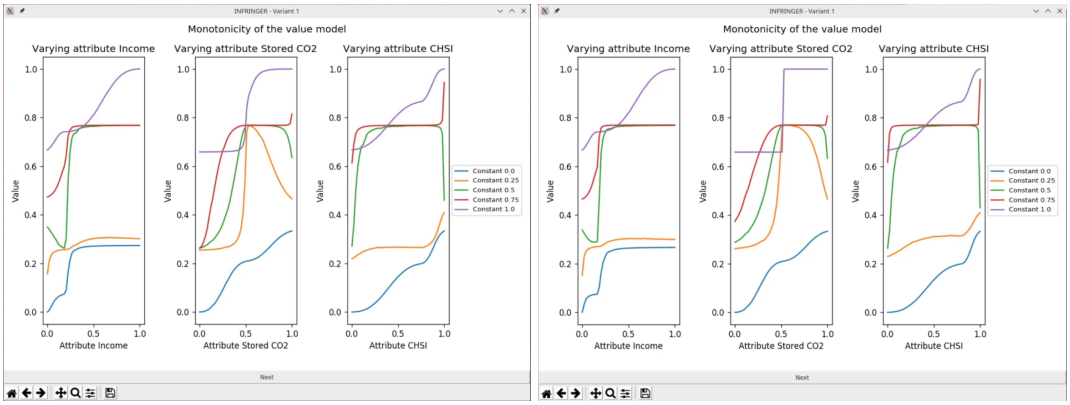
(b) *Ranking of the objective vectors in iteration 2.*

Figure 15: *The ranking of the objective vectors shown to the decision maker during the testing of the first variant of the INFRINGER method. Refer to Figure 14 for an explanation of the plots shown. The best scoring objective vector is indicated by a black diamond in the middle of the black circle.*

The decision maker was puzzled by one of the lines shown in the value function plot and he asked the analyst what it meant. The analyst assured that an explanation exists, but could not give it on the spot.

After inspecting the ranking and value function plots shown, the decision maker was asked again to assess five pairs of Pareto optimal objective vectors, like described in Step 8.5 of Algorithm 1. The decision maker commented that the shown objective vectors seemed to be closer to his preferences, or more focused to what he was looking for, which he thought to be good. The second fitness computed was 40%, which was lower than the fitness computed for the initial BRB model. This might indicate that the model was deteriorating.

Using the pair-wise comparisons, the cost function of the BRB model was updated according to Step 8.7 of Algorithm 1, and the consequent training of the model took around two minutes once again. The resulting value function plot is clearly different from the previous one, and the new ranking, based on the value function, was also noticeably different as can be seen from Figures 15 and 16.



(a) *Value function learned in iteration 1.* (b) *Value function learned in iteration 2.*

Figure 16: *The value functions learned during the testing of the first variant of the INFRINGER method. Refer to Figure 13 for an explanation of the plots shown.*

For a third time, the decision maker was asked to conduct pair-wise comparisons. The resulting fitness was 60%. This time, however, the training of the BRB model was taking a lot of time probably because the solver for finding the optimal parameters of the BRB system was not converging. Observing the cost function values found during the optimization process, the value seemed to vary sporadically between values 0.001 and 0.1. The analyst decided after consulting the decision maker, to stop the training process, which ended the test run of the first variant.

After the first test, the decision maker commented that the learning part part of the method was good, and the second shown best objective vector, shown in the plot with the ranked Pareto optimal objective vectors, was better than the first in his opinion. The decision maker was also happy with this point, and felt that his initial preference for money[7] had changed noticeably during the test. The final solution found for the first variant can be seen as the black diamond in Figure 15(b).

5.2.3 Testing the second variant

Testing of the second variant of the INFRINGER method begun immediately after the testing of the first variant terminated. The same reference point was used as in the first variant, by

[7]Which he used to refer to the total average income.

request of the decision maker. Therefore, actions taken for Steps 1 through 4 in Algorithm 1 were identical to the testing of the first variant.

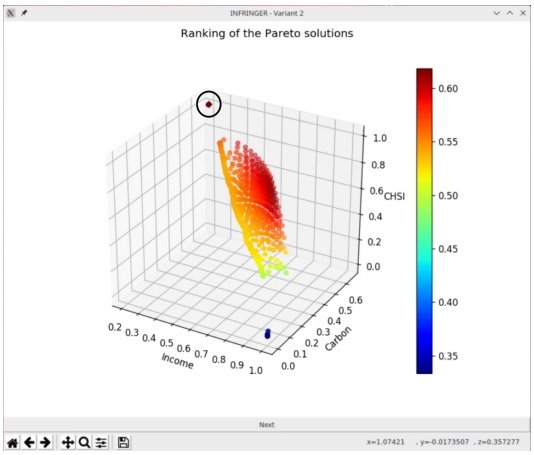
After the first pair-wise comparisons were done according to Step 5 in Algorithm 1, the fitness of the initial model was calculated to be 80% in Step 6 on Algorithm 1. Following Steps 9.1 and 9.2 in Algorithm 1, the first time training of the BRB system took less than a minute, as seen in Table 7.

Next, a plot showing the learned value function, and the ranked Pareto optimal objective vectors, were shown to the decision maker according to Step 9.3 in Algorithm 1. After which a pair-wise comparison was conducted again according to Step 9.4 in Algorithm 1, and the fitness was calculated according to Step 9.5 in Algorithm 1, which was once again 80%. Continuing from Step 9.3 in Algorithm 1, during the second pair-wise comparison, the decision maker made an important remark: he felt that he did not want to give an opinion regarding the 4th pair of objective vectors shown to him, and tabulated in Table 8. Still, the option chosen for the preference was *as good as*, indicating an equal preference by the decision maker.

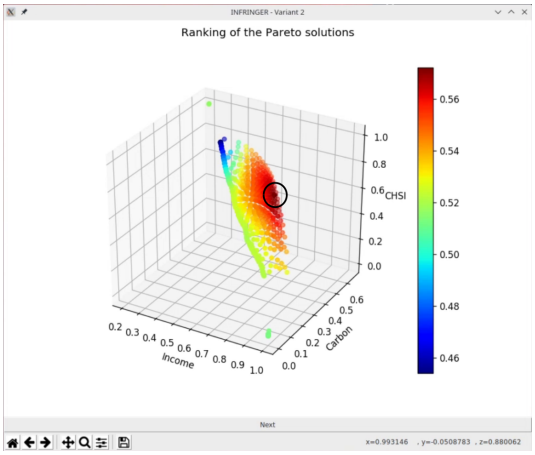
The second time training the model took around one minute, as seen in Table 7, and the resulting value function plot was found to be interesting by the decision maker. More importantly, the best objective vector shown in the ranking plot was noticeably different compared to first iteration. The decision maker noticed that the best vector was similar to the corresponding best vector shown in the second iteration during the testing of the first variant. This was a good thing according to the decision maker.

Iterations continued according to Algorithm 1, and the decision maker was asked four more times to give their preference by conducting pair-wise comparisons. The calculated fitness values varied between 40% and 80% as seen in Table 8, and the training of the model never took more than a minute, which is evident from Table 7. After the fourth iteration, the decision maker felt that the pair-wise comparisons were getting tricky: the compared objective vectors were close in their objective values to each other, and it was hard for the decision maker to give a clear preference which one of the vectors in a pair they preferred.

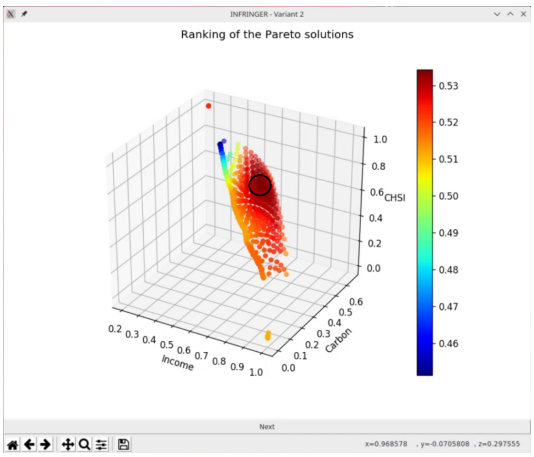
There was also an additional inconsistency in the decision maker's preferences. In the 5th



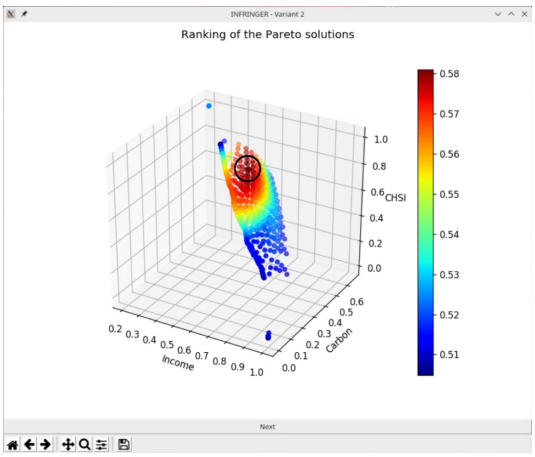
(a) *Ranking of the objective vectors in iteration 1.*



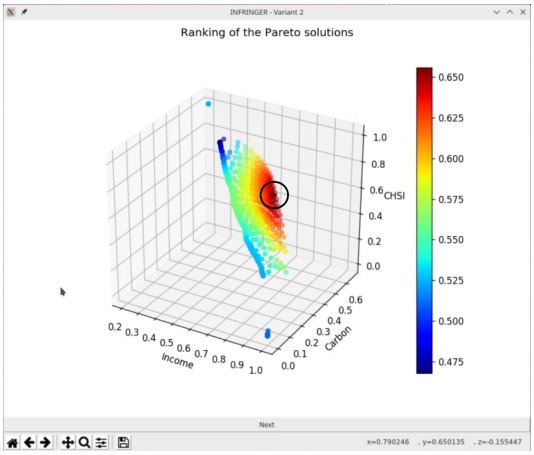
(b) *Ranking of the objective vectors in iteration 2.*



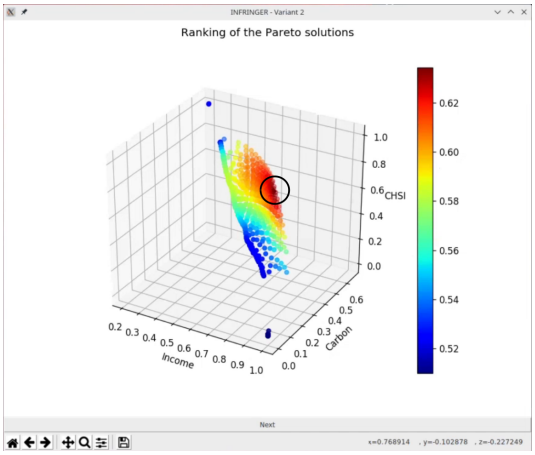
(c) *Ranking of the objective vectors in iteration 3.*



(d) *Ranking of the objective vectors in iteration 4.*

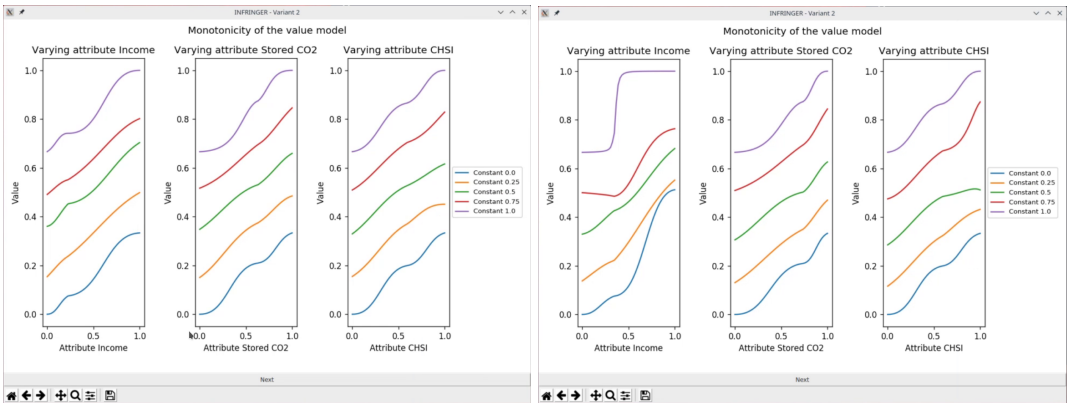


(e) *Ranking of the objective vectors in iteration 5.*



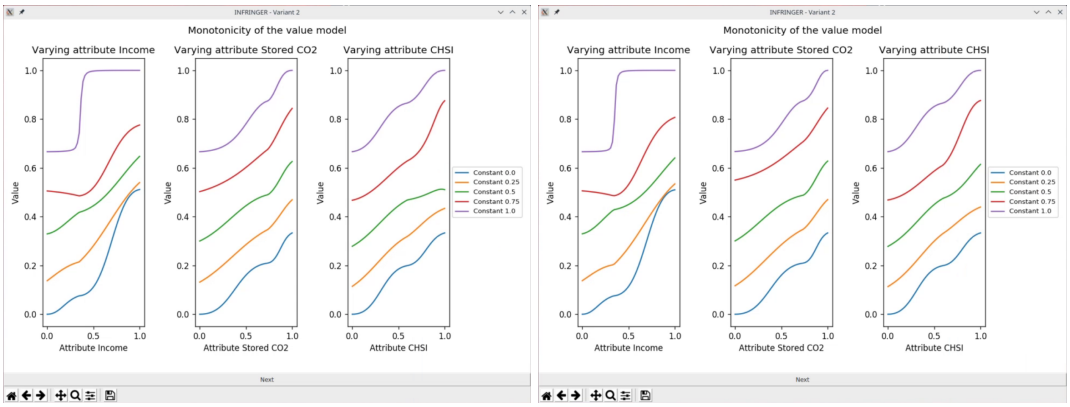
(f) *Ranking of the objective vectors in iteration 6.*

Figure 17: *The ranking of the objective vectors shown to the decision maker during the testing of the second variant of the INFRINGER method. Refer to Figure 14 for an explanation of the plots shown. The best scoring objective vector is indicated by a black diamond in the middle of the black circle.*



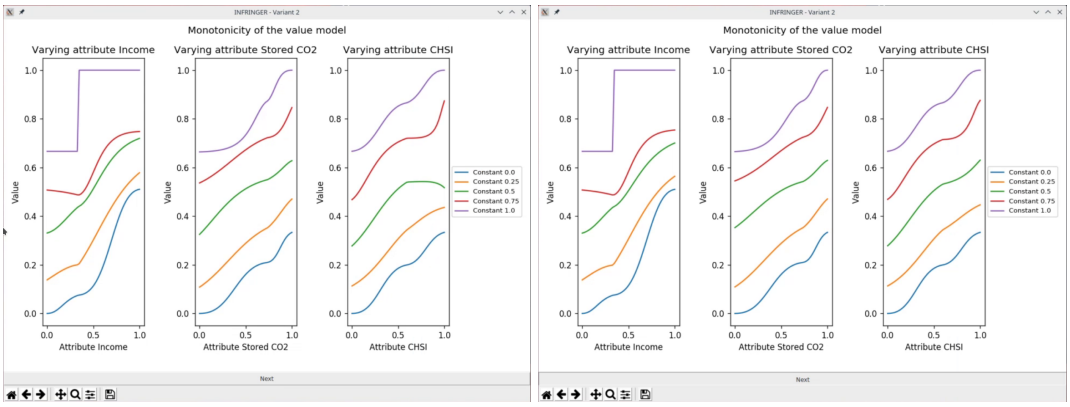
(a) *Value function learned in iteration 1.*

(b) *Value function learned in iteration 2.*



(c) *Value function learned in iteration 3.*

(d) *Value function learned in iteration 4.*



(e) *Value function learned in iteration 5.*

(f) *Value function learned in iteration 6.*

Figure 18: *The value functions learned during the testing of the second variant of the INFRINGER method. Refer to Figure 13 for an explanation of the plots shown.*

Pair	Candidate 1 objective values			Candidate 2 objective values			Preference	Comments
	Income	CO2	CHSI	Income	CO2	CHSI		
Iteration 1 ($\gamma = 0.80$)								
1	4.19e7	7.27e9	3.08e4	4.87e7	7.02e9	2.96e4	=	
2	5.13e7	7.73e9	2.61e4	5.00e7	6.98e9	2.93e4	1	Preference was initially 2, but changed later.
3	5.11e7	7.70e9	2.66e4	5.59e7	6.84e9	2.78e4	1	
4	3.62e7	7.94e9	3.10e4	6.29e7	6.73e9	2.14e4	1	
5	3.62e7	7.94e9	3.10e4	6.29e7	6.73e9	2.14e4	1	Same as pair 4.
Iteration 2 ($\gamma = 0.80$)								
1	4.34e7	7.35e9	3.05e4	5.31e7	6.91e9	2.86e4	1	
2	4.51e7	7.34e9	3.02e4	5.67e7	6.84e9	2.75e4	1	
3	4.22e7	7.51e9	3.06e4	5.87e7	6.85e9	2.66e4	1	
4	1.88e7	8.27e9	3.24e4	6.29e7	6.73e9	2.14e4	=	Decision maker did not want to really give an opinion here.
5	4.50e7	7.88e9	2.84e4	6.28e7	6.75e9	2.18e4	1	
Iteration 3 ($\gamma = 0.80$)								
1	4.31e7	7.44e9	3.05e4	5.61e7	6.84e9	2.77e4	1	
2	5.73e7	7.11e9	2.69e4	5.31e7	6.91e9	2.86e4	1	
3	5.61e7	7.50e9	2.50e4	1.88e7	8.27e9	3.24e4	=	Both far away from what decision maker wanted.
4	4.63e7	7.86e9	2.79e4	3.22e7	7.69e9	3.19e4	1	
5	4.63e7	7.86e9	2.79e4	4.63e7	7.86e9	2.79e4	=	Identical candidates.
Iteration 4 ($\gamma = 0.40$)								
1	5.76e7	6.88e9	2.71e4	4.44e7	7.19e9	3.04e4	2	
2	5.93e7	6.99e9	2.60e4	4.36e7	7.22e9	3.05e4	2	
3	5.31e7	7.23e9	2.82e4	4.29e7	7.24e9	3.06e4	2	
4	4.46e7	7.70e9	2.95e4	3.22e7	7.69e9	3.19e4	1	
5	4.46e7	7.70e9	2.95e4	3.22e7	7.69e9	3.19e4	1	Same as pair 4.
Iteration 5 ($\gamma = 0.60$)								
1	4.60e7	7.75e9	2.87e4	4.80e7	7.68e9	2.83e4	=	Decision maker felt possibly indifferent for pairs 1-3.
2	5.00e7	7.35e9	2.89e4	4.98e7	7.54e9	2.83e4	1	
3	4.94e7	7.07e9	2.94e4	5.21e7	6.95e9	2.88e4	1	Was '=' before the analyst asked for clarification.
4	4.12e7	7.68e9	3.05e4	3.22e7	7.69e9	3.19e4	1	
5	3.22e7	7.69e9	3.19e4	3.22e7	7.69e9	3.19e4	=	
Iteration 6 ($\gamma = 0.60$)								
1	5.63e7	7.15e9	2.72e4	5.08e7	6.97e9	2.91e4	=	
2	4.37e7	7.42e9	3.04e4	5.18e7	6.95e9	2.89e4	1	
3	4.77e7	7.26e9	2.97e4	3.93e7	7.51e9	3.11e4	2	
4	4.63e7	7.86e9	2.79e4	3.22e7	7.69e9	3.19e4	1	
5	4.63e7	7.86e9	2.79e4	3.22e7	7.69e9	3.19e4	1	Same as pair 4.

Table 8: *The preference information given by the decision maker for each iteration while testing the second variant of the INFRINGER method. The equal sign indicates the two candidates were equal preference wise. The fitness after each comparison is also reported for each iteration.*

iteration, the analyst asked for clarification which one of the vectors, in the third pair shown, the decision maker preferred, to which the decision maker replied, that he preferred the first one. However, after analyzing the recorded video of the session, the decision maker had clearly indicated indifference between the pairs, just moments before the analyst asked for clarifications.

A fitness of 100%, which was the fitness threshold γ_{th} set during this case study, was never reached, and the test of the second variant was terminated after six iterations. The decision maker felt that the final best objective vector shown in the ranking plot was, in his words, *approximately* right. The solution found in the second variant of the INFRINGER method can be seen in Figure 17(f).

5.2.4 Concluding remarks

After the tests of both variants of the INFRINGER method were conducted, the decision maker gave a few general comments. He felt that an option to give a preference of indifference should be possible in the pair-wise comparisons of the method. The equal preference option, *as good as*, did not always reflect the decision maker's preferences, which was therefore chosen as a selected option erroneously in some of the pair-wise comparisons. The decision maker felt that the shown best objective vectors in the ranking plots were in his region of interest after the first iteration during the testing of both variants. Additionally, the decision maker felt that the method was partially slow during the training processes where the cost function was minimized; this was especially true during the testing of the first variant.

Overall, the decision maker was happy with the results, even if the end of the INFRINGER method, Steps 10 and 11 in Algorithm 1, were never reached during the testing of both variants. The whole testing process took a total of 47 minutes, after which the video call with the decision maker was terminated.

6 Discussion

In this chapter the viability of the INFRINGER method, which was presented in Chapter 4 will be assessed based on its performance in the case study discussed in Chapter 5. The results produced by the two variants of the INFRINGER method in the case study are compared, and the method's overall performance is briefly and qualitatively contrasted with works in existing literature.

6.1 Viability of the INFRINGER method as a decision support tool

6.1.1 Thoughts on the first variant

Testing the first variant had to be halted after the second iteration due to the cost function of the underlying BRB system not converging to a minimum in a feasible time. The process of minimizing the cost function had to be stopped by the analyst.

Observing the value function plots in Figure 16, it is evident that the resulting value function is not monotonic, and the assumptions made in Chapter 2 made for the value function do not hold. There are three plausible explanations for the value function learned not to be monotonic.

The first explanation for the learned value function not to be monotonic, is that the cost function (4.11) formulated to enforce the monotonicity is not appropriate. The cost function does not cover all possible values for the three objectives present in the objective function. Therefore, the cost function allows for the value function to breach monotonicity in areas where the monotonicity is not tested for. Since the optimal value found for the aggregated cost function (3.30) was never truly zero, as evident in Table 7, it is also possible that the optimal value found for the aggregate cost function consisted mostly of the monotonic cost function's possible non-zero value, meaning the BRB system was simply not able to model a monotonic value function.

The second explanation for the learned value function to not be monotonic, is the possibility that the assumptions made regarding the decision maker, or the cost function, are not true.

It is possible that the decision maker was not rational, and therefore the assumption of a monotonic value function was wrong. This in turn made it impossible for the monotonic cost function to be zero. Alternatively, the assumption that a cost function can be found, which models the preferences of a decision maker could be false. These explanations may also manifest themselves in the cost function regarding the scores given by the decision maker (4.13) or the cost function regarding the pair-wise comparisons (4.15), which were also both relevant during the minimization of the aggregated cost function. The assumption made regarding the pair-wise cost function related to the pair-wise comparisons is also supported by the observation where the decision maker had expressed a change in his preferences for the total average income during the case study. This change of preferences during the INFRINGER method is not accounted for in the additive update strategy of the pair-wise cost function, where it was assumed that the preferences of the decision maker do not change.

It is also possible that the cost function related to the scores given by the decision maker imposed restrictions on the BRB system, such that the aggregated cost function could not be minimized. Lowering the δ value in (4.13) could have alleviated this effect.

However, after the test the decision maker felt like the best objective vector suggested in the second iteration was close to what he would have deemed preferable. The decision maker felt like some learning had happened. Indeed, observing Figure 16 it is clear that the value function has changed between iterations, and observing Figure 15 it is evident that the best objective vector indicated by a black diamond and surrounded by a black circle has moved between iterations. This would also suggest that the underlying value function has changed. Because of the decision maker's remark and the observation of both the best objective vector changing and the plotted value function changing, it is concluded that the underlying BRB system did learn a value function reflecting the decision maker's preferences.

6.1.2 Thoughts on the second variant

The testing of the second variant had to be also stopped by the analyst. After six iterations, the fitness threshold was never reached, and the decision maker got tired. The analyst and decision maker agreed to stop the method after the sixth iteration.

Not reaching the fitness threshold set might have been due to the decision maker getting tired, as is evident in Table 8 in the comparison of pair 2 during iteration 5: the decision maker gave a different preference concerning the second pair, when asked for his preference a moment later he had given his initial preference. This suggests that the decision maker might have been more and more inconsistent also in his other decisions, even if not as evident as in the case of the aforementioned pair-wise comparison. Therefore, from the possibility for the decision maker to have become inconsistent because of tiredness in conjunction with the fact that the INFRINGER method expects consistent preference information from the decision maker, it could have lead to the possibility that no feasible value function exists for modelling the preferences of the decision maker if the assumption made regarding the value function are assumed to be true. Moreover, a possible future research topic could be testing for the consistency of the preferences of a decision maker.

Regardless, the value function learned in the second variant was noticeably better than in the first variant. This is evident when observing Figure 18, where the learned value function are, for the most part, monotonically increasing, and noticeably different between iterations. Additionally, observing the rankings of the objective vectors in the second variant seen in Figure 17, it seems that the best objective vector is converging – the differences between its position in plots between consequent iterations is getting smaller. Based on these observations, it is concluded that learning is happening in the second variant as well. The reason why the second variant was able to learn a better value function could be because it was simpler – only pair-wise comparisons were required from the decision maker, therefore, imposing a lighter cognitive load on the decision maker compared to the first variant.

It was also observed that the minimization of the cost function during the second variant was a lot faster as can be seen from Table 7. Training never took more than two minutes suggesting that the second variant could be more usable in a real life application, where excessive waiting is not desired.

The decision maker was happy with the results of the second variant, even if the method had to be halted before the last Steps 10 and 11 in Algorithm 1. Nevertheless, the suggested best objective vector was in the region of interest of the decision maker after the first few iterations.

Variant	Income	CO2	CHSI
1	4.16e7	7.94e9	2.94e4
2	4.63e7	7.86e9	2.79e4

Table 9: *The best objective vectors according to the learned value function in the case study produced by both variants of the INFRINGER method. The vectors are taken from the partial final partial ranking calculated in each variant because neither of the variants was able to reach the last steps of Algorithm 1.*

6.1.3 General remarks

Even though neither of the variants reached the end of the INFRINGER method, the most recent best objective vector in both variants, in terms of the score given by value function modelled by the BRB system, can be extracted from Tables 6 and 8: the first candidate in the fourth pair is always chosen to be the objective vector with the highest score, as described in Section 4.1.4. The two best objective vectors extracted are presented in Table 9.

Observing Table 9, it is clear the the suggested best objective vectors in both variants were close to each other. This is also somewhat evident from the ranking produced in the final iteration of both variants as seen in Figures 15(b) and 17(f). The rankings are similar, with the values of the objective vectors increasing in the direction of increased CO2[1]. Based on the similarity of the best objective vectors suggested by both variants, it is concluded that both variants were able to learn a similar value function for the same decision maker. This suggests that the INFRINGER method works with definite intent behind it instead of just expressing random behavior.

A recurring theme during the case study was a need for distinction between equal preference and no preference. As is seen from the comments-column in both Tables 6 and 8, there were multiple occasions during the testing of both variants where the decision maker felt indifferent or did not want to give an opinion when tasked with comparing a pair of objective vectors[2]. The learned value function does not capture indifference as a kind of preference,

[1] Carbon in the figure.

[2] Dubbed as Candidates in the tables.

and during the case study, indifference was synonymous to equal preference. This is partly caused by an oversight by the author of this thesis and should be addressed in future iterations of the INFRINGER method.

Another note worthy theme evident from Tables 6 and 8 is the presence of duplicate objective vectors present in a pair or between two different pairs during the pair-wise comparisons in the same iteration. This phenomenon is not necessarily bad. The consistency of the decision maker can be checked in the comparison of candidates like in iteration 4 in Table 8, where both pairs 4 and 5 were identical. It can be seen that the decision maker gave the same preference for both pairs; this is an indication of a consistent decision maker and is a recurring phenomena in other similar comparisons throughout the iterations of both variants. It is therefore concluded that the decision maker was mostly consistent and the oversight in the comparison of pair 3 during iteration 5 shown in Table 8 was most likely due to tiredness. However, the decision maker was still inconsistent to some degree, because he expressed that his preferences had changed after the testing of the first iteration. This comment from the decision maker cannot be ignored and must be accounted for.

However, the method for choosing the objective vectors to be compared in both variants of the INFRINGER method should be still refined, even if duplicate vectors in pairs could function as a way to check the consistency of the decision maker in expressing his/her preferences. Comparison of identical pairs is redundant and conveys no new information useful in learning the decision maker's value function. The aforementioned consistency check of the decision maker's preferences emerging from the comparison of identical pairs, was mostly due to sheer luck and was not an intended feature of the method. Therefore, the method for choosing the pairs to be compared should be reworked. A way to check for the consistency of the decision maker can then be implemented separately with clearer intent.

The method for choosing the objective vectors to be scored by the decision maker in the first variant is based on similar principles as the choice of the pairs in the pair-wise comparisons. This could lead to similar problems to what were encountered with choosing the pairs for the pair-wise comparisons. Therefore, it is suggested that the way for choosing the objective vectors to be scored should also be refined.

Different times elapsed during the minimization of the cost function are shown in Table 7 for both variants of the INFRINGER method alongside with the optimal value found for the cost function. The elapsed times, and optimal cost function values, are noticeably lower in the second variant, which indicates that the second variant may be more suitable in eliciting the decision maker's preferences, considering the fact that both variants produced a similar best objective vector after their corresponding last iterations.

Lastly, one obvious remark remains to be discussed. Many of the choices, especially regarding the structure of the BRB system in the INFRINGER method, have been mostly a result of a pragmatic approach consisting of trial and error. The author of this thesis argues that the chosen parameters discussed in Chapter 4 offer a good trade-off between prediction and computational efficiency of the BRB system. However, these claims cannot be backed up quantitatively. This can be excused, if the method is treated as a proof of concept, which it is.

6.2 A brief qualitative comparison of the INFRINGER method to other similar methods

Existing methods in literature, which try to learn a preference model of a decision maker were briefly presented in Section 1.3. Albeit comparing the INFRINGER method directly to existing methods is impossible due to its novel nature and lack of quantitative results, a brief qualitative comparison can still be conducted.

Common between the methods mentioned in Section 1.3 and the INFRINGER method, is the use of pair-wise comparisons to elicit preference information from a decision maker. Another common factor is the use of value functions to model the preferences. All the methods strive to aid the decision maker(s) reach a satisfying decision, which is also common to the INFRINGER method. Lastly, the interactive nature of the methods seems to be a common factor as well, suggesting that the best way to elicit preference information from a decision maker should be conducted in situ.

The belief rule-based system the INFRINGER method is based on offers the potential for producing explainable results regarding the learned preference model for a decision maker.

While this thesis does not dive into the details of explainable machine learning models, it can be shown that belief-rule based systems can offer explainable results [16, 17]. The potential for explainability, and use of rule-based belief systems, are the two main factors which differentiate the INFRINGER method from existing methods in literature.

All methods in literature, and the INFRINGER method, have their own set of test of problem(s), which are used to assess the method's capability to learn a decision maker's preference model. Test problems may have been tailored, or chosen appropriately fitting the specific needs and features of the tested method, and a test problem that performs very well with one method, might perform poorly with another method. The need to compare different methods for learning the preferences of a decision maker, calls for a uniform testing framework to be developed. This framework should contain established problems, and artificial[3] decision makers with preference models known a priori. This kind of framework could be used to assess different methods in their ability of learning the preference model of a decision maker, and to produce comparable, quantitative results. To the knowledge of the author of this thesis, this kind of framework does not yet exist.

[3] As in having either simulated or predefined preference behavior.

7 Conclusions

Based on the discussion in Chapter 6 regarding the case study, it is concluded that the INFRINGER method could be a plausible tool to aid a decision maker to reach a satisfying decision in a multi-objective optimization problem. The method also manifested promising results in modelling the preferences of the decision maker in the case study, as is evident from not only the consistency of the method between the two tested variants in predicting a similar best objective vector, but is also evident from the fact that the predicted objective vectors were both close to what the decision maker was aiming for. However, many of the choices made in the development of the INFRINGER method were purely based on trial and error. It could be possible to significantly improve the performance of the INFRINGER method by conducting a more through analysis on the effect of the different parameters in the BRB system have on the performance of the method.

The explainability of the results of the INFRINGER method in modelling a preference model were not explored in this thesis to an extent were a conclusion could be drawn. However, the explainable nature of the belief-rule based systems used in the INFRINGER method was established, which leads to the notion that the INFRINGER method could be further developed to produce explainable results.

In conclusion, the research question 1, 2, and 3 listed in Section 1.4 can be answered affirmatively, or at least with a very strong *perhaps*. Question 4 remains unanswered in this thesis, but the existing literature suggests that it is very plausible that the resulting preference model is explainable. If any future research is conducted based on the INFRINGER method or a method similar to it, a special emphasis should be put on the study of the model's ability to produce explainable results. It is also evident that some methodology is needed for comparing methods that aim to learn a decision maker's preferences. The author of this thesis suggests that a testing framework should be developed for this purpose.

Acknowledgements

In no particular order.

The author of this thesis would like to thank Jian-bo Yang, Ling Xu, and Swati Sachan for the insightful discussion sessions and mentoring over Skype related to belief rule-based systems.[1] The author would also like to thank his supervisors Kaisa Miettinen and Jussi Hakanen for their guidance and feedback related to the writing of this thesis. And last, but not least, thanks for Kyle Eyvindson for acting as the decision maker in the case study.

This thesis is related to the thematic research area DEMO: Decision Analytics utilizing Causal Models and Multiobjective Optimization of the University of Jyväskylä[2], and has been funded by the Academy of Finland (project number: 322221 in 2019-2023).

[1] From the University of Manchester.

[2] `https://www.jyu.fi/demo`

Bibliography

[1] K. Miettinen. *Nonlinear multiobjective optimization*. Kluwer Academic Publishers, 1999.

[2] J. Teghem, D. Dufrane, M. Thauvoye, and P. Kunsch. "Strange: an interactive method for multi-objective linear programming under uncertainty". *European Journal of Operational Research* 26(1), 1986, 65–82.

[3] K. Miettinen and M. M. Mäkelä. "Synchronous approach in interactive multiobjective optimization". *European Journal of Operational Research* 170(3), 2006, 909–922.

[4] K. Miettinen, P. Eskelinen, F. Ruiz, and M. Luque. "NAUTILUS method: an interactive technique in multiobjective optimization based on the nadir point". *European Journal of Operational Research* 206(2), 2010, 426–434.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II". *IEEE Transactions on Evolutionary Computation* 6(2), 2002, 182–197.

[6] M. Öztürké, A. Tsoukiás, and P. Vincke. "Preference modelling". *International Series in Operations Research & Management Science*. Vol. 78. Springer New York, 2005, 27–59.

[7] M. Aggarwal. "Learning of a decision-maker's preference zone with an evolutionary approach". *IEEE Transactions on Neural Networks and Learning Systems* 30(3), 2019, 670–682.

[8] R. de Almeida, G. Reynoso-Meza, and M. T. A. Steiner. "Multi-objective optimization approach to stock market technical indicators". *Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, 3670–3677.

[9] L. G. More, M. A. Brizuela, H. L. Ayala, D. P. Pinto-Roa, and J. L. V. Noguera. “Parameter tuning of CLAHE based on multi-objective optimization to achieve different contrast levels in medical images”. *Proceedigns of the 2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2015, 4644–4648.

[10] R. Marler and J. Arora. “Survey of multi-objective optimization methods for engineering”. *Structural and Multidisciplinary Optimization* 26(6), 2004, 369–395.

[11] M. Mönkkönen, A. Juutinen, A. Mazziotta, K. Miettinen, D. Podkopaev, P. Reunanen, H. Salminen, and O.-P. Tikkanen. “Spatially dynamic forest management to sustain biodiversity and economic returns”. *Journal of Environmental Management* 134, 2014, 80–89.

[12] C. Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. *Nature Machine Intelligence* 1(5), 2019, 206–215.

[13] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2011.

[14] J.-B. Yang, J. Liu, J. Wang, H.-S. Sii, and H.-W. Wang. “Belief rule-base inference methodology using the evidential reasoning Approach-RIMER”. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 36(2), 2006, 266–285.

[15] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. *Nature* 521(7553), 2015, 436–444.

[16] S. Sachan, J.-B. Yang, and D.-L. Xu. “Comparing strategies on explainability of machine learning models with belief-rule-based expert systems: a case study on lending decisions”. 10th Annual European Decision Sciences Conference. 2019.

[17] S. Sachan, J.-B. Yang, D.-L. Xu, D. Benavides, and Y. Li. “An explainable AI decision-support-system to automate loan underwriting”. *Expert Systems with Applications* 144, 2019, 113100.

[18] M. T. Ribeiro, S. Singh, and C. Guestrin. “"Why should I trust you?"” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD ’16*. ACM Press, 2016.

[19] R. Battiti and A. Passerini. “Brain–computer evolutionary multiobjective optimization: a genetic algorithm adapting to the decision maker”. *IEEE Transactions on Evolutionary Computation* 14(5), 2010, 671–687.

[20] M. Aggarwal, J. Heinermann, S. Oehmcke, and O. Kramer. “Preferences-based choice prediction in evolutionary multi-objective optimization”. *EvoApplications 2017: Applications of evolutionary computation*. Springer, 2017, 715–724.

[21] J. Branke, S. Greco, R. Slowinski, and P. Zielniewicz. “Learning value functions in interactive evolutionary multiobjective optimization”. *IEEE Transactions on Evolutionary Computation* 19(1), 2015, 88–102.

[22] I. Dewancker, M. McCourt, and S. Ainsworth. *Interactive preference learning of utility functions for multi-objective optimization*. 2016. arXiv: `1612.04453 [math.OC]`.

[23] A. Bemporad and D. Piga. *Active preference learning based on radial basis functions*. 2019. arXiv: `1909.13049 [cs.LG]`.

[24] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of multiobjective optimization*. Academic Press, 1985.

[25] W. H. Press. *Numerical recipes 3rd edition: the art of scientific computing*. Cambridge University Press, 2007.

[26] J. Nocedal. *Numerical optimization*. Springer, 2006.

[27] J. Forrest, T. Ralphs, S. Vigerske, H. Lou, B. Kristjansson, Jpfasano, E. Straver, M. Lubin, H. G. Santos, Rlougee, and M. Saltzman. *Coin-or/cbc: Version 2.9.9*. 2018.

[28] M. Pescador-Rojas, R. H. Gómez, E. Montero, N. Rojas-Morales, M.-C. Riff, and C. A. C. Coello. "An overview of weighted and unconstrained scalarizing functions". *Evolutionary Multi-Criterion Optimization*. Springer, 2017, 499–513.

[29] J. T. Buchanan. "A naive approach for solving MCDM problems: the GUESS method". *Journal of the Operational Research Society* 48(2), 1997, 202–206.

[30] I. Das and J. E. Dennis. "Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems". *SIAM Journal on Optimization* 8(3), 1998, 631–657.

[31] K. Miettinen and M. M. Mäkelä. "On scalarizing functions in multiobjective optimization". *OR Spectrum* 24(2), 2002, 193–213.

[32] S. Bechikh, L. B. Said, and K. Ghedira. "Estimating nadir point in multi-objective optimization using mobile reference points". *IEEE Congress on Evolutionary Computation*. IEEE, 2010.

[33] K. Deb, K. Miettinen, and S. Chaudhuri. "Toward an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches". *IEEE Transactions on Evolutionary Computation* 14(6), 2010, 821–841.

[34] B. Xin, L. Chen, J. Chen, H. Ishibuchi, K. Hirota, and B. Liu. "Interactive multiobjective optimization: a review of the state-of-the-art". *IEEE Access* 6, 2018, 41256–41279.

[35] A. B. Ruiz, K. Sindhya, K. Miettinen, F. Ruiz, and M. Luque. "E-NAUTILUS: A decision support system for complex multiobjective optimization problems based on the NAUTILUS method". *European Journal of Operational Research* 246(1), 2015, 218–231.

[36] Y.-W. Chen, J.-B. Yang, D.-L. Xu, Z.-J. Zhou, and D.-W. Tang. “Inference analysis and adaptive training for belief rule based systems”. *Expert Systems with Applications* 38(10), 2011, 12845–12860.

[37] X. Xu, X. Yan, C. Sheng, C. Yuan, D. Xu, and J. Yang. “A belief rule-based expert system for fault diagnosis of marine diesel engines”. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50, 2 2017, 1–17.

[38] G. Kong, D.-L. Xu, J.-B. Yang, X. Yin, T. Wang, B. Jiang, and Y. Hu. “Belief rule-based inference for predicting trauma outcome”. *Knowledge-Based Systems* 95, 2016, 35–44.

[39] D.-L. Xu, J. Liu, J.-B. Yang, G.-P. Liu, J. Wang, I. Jenkinson, and J. Ren. “Inference and learning methodology of belief-rule-based expert system for pipeline leak detection”. *Expert Systems with Applications* 32(1), 2007, 103–113.

[40] Y.-W. Chen, S.-H. Poon, J.-B. Yang, D.-L. Xu, D. Zhang, and S. Acomb. “Belief rule-based system for portfolio optimisation with nonlinear cash-flows and constraints”. *European Journal of Operational Research* 223(3), 2012, 775–784.

[41] G. Kong, D.-L. Xu, R. Body, J.-B. Yang, K. Mackway-Jones, and S. Carley. “A belief rule-based decision support system for clinical risk assessment of cardiac chest pain”. *European Journal of Operational Research* 219(3), 2012, 564–573.

[42] P. Korhonen and J. Wallenius, eds. *Multiobjective optimization: interactive and evolutionary approaches*. Springer Berlin Heidelberg, 2008. Chap. 8, 195–212.

[43] A. V. Lotov and K. Miettinen, eds. *Multiobjective optimization: interactive and evolutionary approaches*. Springer Berlin Heidelberg, 2008. Chap. 9, 213–244.

[44] P. S. Foundation. *Python language reference, version 3.8.* 2020. URL: `https://www.python.org` (visited on 02/23/2020).

[45] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, and P. van Mulbregt. "SciPy 1.0: fundamental algorithms for scientific computing in Python". *Nature Methods* 17, 2020, 261–272.

[46] S. van der Walt, S. C. Colbert, and G. Varoquaux. "The NumPy array: a structure for efficient numerical computation". *Computing in Science & Engineering* 13(2), 2011, 22–30.

[47] J. D. Hunter. "Matplotlib: a 2D graphics environment". *Computing in Science & Engineering* 9(3), 2007, 90–95.

[48] J. Rasinmäki, A. Mäkinen, and J. Kalliovirta. "SIMO: an adaptable simulation framework for multiscale forest resource data". *Computers and Electronics in Agriculture* 66(1), 2009, 76–84.

[49] W. E. Hart, J.-P. Watson, and D. L. Woodruff. "Pyomo: modeling and solving mathematical programs in Python". *Mathematical Programming Computation* 3(3), 2011, 219–260.

[50] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola. *Pyomo — optimization modeling in Python*. Springer, 2017.

[51] J. Malmberg. "Data-driven interactive multiobjective optimization using cluster based surrogate in discrete decision space". MA thesis. University of Jyväskylä, 2018.

Appendices

A Code example of using the BRB software framework developed in this thesis

```python
import numpy as np
import matplotlib.pyplot as plt
from brb import BRB

def himmelblau_example():
    def himmelblau(x):
        """The himmelblau function.

        """
        x = np.squeeze(x)
        res = (x[:, 0] ** 2 + x[:, 1] - 11) ** 2 + (
            x[:, 0] + x[:, 1] ** 2 - 7
        ) ** 2
        return res.reshape(1, -1, 1)

    def linspace2d(low, up, n):
        """Used to create evenly distributed sparce 2D coordinate points.

        """
        step_s = (up - low) / n
        return (
            np.mgrid[
                low[0] : up[0] + 0.1 : step_s[0],
                low[1] : up[1] + 0.1 : step_s[1],
            ]
            .reshape(2, -1)
            .T
        )

    # Define the precedent referential values
    refs = np.array([[-6, -4, -2, 0, 2, 4, 6], [-6, -4, -2, 0, 2, 4, 6]])
```

```
    # Define the consequent referential values
    consequents = np.array([[0, 200, 500, 1000, 2200]])

    # Construct an initial model
    brb = BRB(refs, consequents, f=himmelblau)

    # Generate a random set of inputs and outputs
    low = np.array([-6, -6])
    up = np.array([6, 6])
    n = 8
    xs_train = linspace2d(low, up, n)
    ys_train = himmelblau(xs_train)

    # Real data to compare to
    xs = linspace2d(low, up, 14)
    ys = himmelblau(xs)

    # Untrained data
    res = brb.predict(xs)
    ys_untrained = np.sum(
        res.consequents * res.consequent_belief_degrees, axis=1
    )

    # Train the BRB
    brb.train(xs_train, ys_train, brb._flatten_parameters())

    # trained data
    res = brb.predict(xs)
    ys_trained = np.sum(
        res.consequents * res.consequent_belief_degrees, axis=1
    )

    # Plot the results
    ys = np.squeeze(ys)
    plt.plot(np.linspace(0, len(ys), len(ys)), ys, label="The Himmelblau
     ↪  function")
```

```
67      plt.plot(np.linspace(0, len(ys), len(ys)), ys_untrained,
        ↪  label="Untrained")
68      plt.plot(np.linspace(0, len(ys), len(ys)), ys_trained,
        ↪  label="Trained")
69      plt.xlabel("arb")
70      plt.ylabel("$f(x_1, x_2)$")
71      plt.title("Trained vs untrained BRB system for predicting the
        ↪  Himmelblau function")
72      plt.legend()
73      plt.show()
74
75
76  if __name__=="__main__":
77      himmelblau_example()
```