# Preferences in Case-Based Reasoning


Amira Abdel-Aziz

Fachbereich Mathematik und Informatik

Philipps-Universität Marburg

**Dissertation**

zur Erlangung eines

*Doktorgrades der Naturwissenschaften (Dr. rer. nat.)*

Marburg/Lahn, 2016

Vom Fachbereich Mathematik und Informatik der Philipps-Universität Marburg
Hochschulkennziffer (1180) als Dissertation am 30. September 2016 angenommen.

1. Gutachter Prof. Dr. Eyke Hüllermeier, Universität Paderborn

2. Gutachter Prof. Dr. Mirjam Minor, Universität Frankfurt

Tag der Einreichung: 28. Juni 2016.
Tag der mündlichen Prüfung: 7. Oktober 2016.

# Abstract

Case-based reasoning (CBR) is a well-established problem solving paradigm that has been used in a wide range of real-world applications. Despite its great practical success, work on the theoretical foundations of CBR is still under way, and a coherent and universally applicable methodological framework is yet missing. The absence of such a framework inspired the motivation for the work developed in this thesis. Drawing on recent research on preference handling in Artificial Intelligence and related fields, the goal of this work is to develop a well theoretically-founded framework on the basis of formal concepts and methods for knowledge representation and reasoning with preferences.

A preference-based approach to CBR appears to be appealing for several reasons, notably because case-based experiences naturally lend themselves to representations in terms of preference relations, even when not dealing with preference information in an explicit way. Moreover, the flexibility and expressiveness of a preference-based formalism well accommodate the uncertain and approximate nature of case-based problem solving.

Preference-based CBR is conceived as a case-based reasoning methodology in which problem solving experience is mainly represented in the form of contextualized preferences, namely preferences for candidate solutions in the context of a target problem to be solved. The work in this thesis is based on the idea of the formalization of a preference-based CBR framework, by embedding a method to predict a most plausible candidate solution given a set of preferences on other solutions, deemed relevant for the problem at hand. In this framework, case-based problem solving is formalized as a search process, in which a solution space is traversed through the application

of adaptation operators, and the choice of these operators is guided by case-based preferences.

For further optimization of the developed framework, we include methods that learn how to combine given local similarity measures into a global one, for ensuring optimal performance of the case-based reasoning system. The learning method for the solution space is based on the Bayesian approach for distance metric learning, and the learning of the similarity measures in the problem space has been reduced to binary classification by a classification method (in our learning we use the perceptron algorithm). To complete the framework and maintain its efficiency, methods for dynamic case base maintenance specifically suited for our preference-based CBR (Pref-CBR) framework are also embedded. The main goal of these maintenance methods is to increase efficiency of case-based problem solving, by reducing the size of the case base, while maintaining performance of the system. We implemented some experiments which show the efficacy of our Pref-CBR problem-solving framework. We also show several applications which highlight in isolation the affect of our methods for learning similarity measures on the performance of the search, as well as the affect of our maintenance strategies on the efficiency of the search.

## Zusammenfassung

Das Paradigma des fallbasierten Schließens (engl.: Case-Based Reasoning, CBR) hat sich in zahlreichen Anwendungen als Werkzeug zur Lösung neuer Probleme mithilfe bereits gelöster Probleme bewährt. Trotz dieses Erfolgs mangelt es dem CBR noch wie vor an theoretischen Grundlagen sowie einem allgemeinen methodischen Unterbau. Dies bildet den Ausgangspunkt für die vorliegende Arbeit. Das Hauptanliegen ist die Entwicklung eines theoretisch fundierten Konzeptes, in dem die formale Wissensrepräsentation durch Präferenzrelationen erfolgt. Präferenzen und paarweise Vergleiche bieten einen sehr natürlichen Zugang zur Modellierung und Durchführung von Problemlöseprozessen, und ihre Nutzung ist daher von großem Interesse für die künstlichen Intelligenz und angrenzende Bereiche.

Die Einbeziehung von Präferenzen in das fallbasierte Schließen ist aus verschiedenen Gründen vorteilhaft. Insbesondere können Problemlösungsprozesse durch sukzessive Anwendung von qualitativen Alternativentscheidungen modelliert werden. Auf diese Weise wird Erfahrungswissen über zurückliegende Entscheidungen in ähnlichen Situationen berücksichtigt. Dieser intuitive Ansatz spiegelt den Annäherungscharakter von Problemlösungsprozessen gut wider und ermöglicht entlang von Entscheidungspfaden auch einen flexiblen Umgang mit unsicheren Informationen.

Präferenzbasiertes fallbasiertes Schließen nutzt eine Falldatenbank, in der Problemlösungserfahrung in Form von kontextbezogenen Präferenzen abgelegt wird. Neue Probleme werden durch Wiederverwendung von Präferenzpfaden ähnlicher bereits gelöster Probleme gelöst. In dieser Arbeit wird der formale Rahmen für eine solche Vorgehensweise entwickelt. Problemlösung erfolgt als Suche im Lösungsraum, wobei Lösungen durch die Anwendung von Operatoren angepasst und verbessert werden, deren Auswahl durch ein Maximum-Likelihood-Modell mit den fallbasierten Präferenzen erfolgt.

Zur weiteren Verbesserung des vorgestellten Problemlösungsverfahrens werden Techniken zur Anpassung von Ähnlichkeitsmaßen entwickelt. Ähnlichkeitsmaße beziehen sich einerseits auf den Vergleich von Problemen untereinander und andererseits von Lösungen. Die Anpassung erfolgt durch Lernen einer für den Problemlösungsvorgang optimalen Gewichtung derjenigen Aspekte, deren Unterschiede für Vergleiche herangezogen werden. Für den Raum der Probleme ergibt sich die Gewichtung als Ergebnis eines mit dem Perzeptronalgorithmus gelösten Klassifikationsproblemes von Problemen, deren Ähnlichkeiten mit Lösungspräferenzen kompatibel sein sollen. Die Wichtung des Lösungsraumes ergibt sich dagegen aus einem Wahrscheinlichkeitsmodell durch Anwendung des Satzes von Bayes. Abschließend werden verschiedene Techniken zur kontinuierlichen Wartung der Falldatenbank vorgestellt, die durch Ausdünnung von Fällen und Präferenzen bei gleichbleibender Problemlösungsqualität auf eine Steigerung der Effizienz zielen.

I would like to dedicate this doctoral dissertation to my father. He always believed in me and continuously encouraged and motivated me, without his kind advice and support I would not have been able to continue this process. He is and always will be my great role model. The same goes with my dear mother who always helped me and gave me advice and taught me how to be strong and responsible since I was a child. My mother has always been a source of strength, power and the light which always directed me through dark tunnels.

I would also like to dedicate this to my dear husband, who made many sacrifices throughout the past years to let me finish my work. I thank you deeply from my heart and I appreciate everything you have done for me.

I dedicate this dissertation also to my lovely children, who helped me so much at many times and always cared so much for me. No words can express my gratitude and love for you.

I dedicate this work also to my dear brother, who has always been there for me and will always be a part of my heart.

# Acknowledgements

I would like to express my appreciation and special thanks to my advisor Prof. Dr. Eyke Hüllermeier, you have been a tremendous mentor for me. I would like to thank you for encouraging me to continue on with my research, and for allowing me to grow as a scientist. Your ideas and advice to me have been priceless and I am honored to have had the chance to work with you. I would also like to thank my committe members, Prof. Dr. Mirjam Minor, and Prof. Dr. Bernd Freisleben for giving me the time to read my work and be there for me. I would also like to thank my department members for always offering help and support at times when I needed them.

I would also like to sincerely thank the Yousef Jameel program for giving me the chance to produce this work. They have been supporting me throughout the past four years and I thank them greatly for their trust in me and for the opportunity they have provided to me to pursue and finish my PhD research. I appreciate very much your help and support and I hope that I have been up to your expectations.

I dearly thank my husband and my beautiful children for all the sacrifices they have made on my behalf, I am truly lucky to have such a wonderful family and nothing that I can say would express how much I love them. I would like to express special appreciation to my husband for supporting me and encouraging me to continue at difficult times. His encouraging words and confidence in me kept me going. I cannot thank enough my kids, who have also helped me and encouraged me. They were responsible enough to take care of many things to help me focus on my work.

A special thanks to my family, words cannot even express how grateful I am to my father, my mother and my brother for being there for me, and I am blessed to have such a wonderful family. Thank you so much for all

your kind support and encouragement at all times. I would also like to greatly thank my mother and father, who are role models for me. Thinking about what they achieved in life, even through hardships, always gave me the strength to continue and reach my goals in life.

I would also like to thank all of my friends for helping me in many ways, and motivating me to strive towards my goals. My appreciation and many thanks go to Dr. Marc Strickert, with whom I have worked closely and learned so much from. He was always there whenever I needed help and he has been a great source of support and encouragement throughout my PhD research trip. Last but not least, I would like to specifically thank Miriam Gross for her help and support from the first day I came here to Germany and even before that, throughout all my arrangements for coming here. No words can really express how grateful I am to all what she has done for me.

# Contents

# List of Figures

## LIST OF FIGURES

# List of Tables

# 1

# Introduction

Case-based reasoning (CBR) is a problem solving paradigm built upon a rule of thumb suggesting that similar problems tend to have similar solutions [1]. More specifically, the idea of CBR is to exploit experience from similar problems in the past and to adapt successful solutions to the current problem. Thus, the core of every case-based problem solver is the case base, which is a collection of memorized chunks of experience, called cases.

The field of CBR arose originally from the research in cognitive psychology, where CBR research was highly regarded as a plausible high-level model for cognitive processing [2]. The idea of CBR is intuitively appealing because it is similar to human problem-solving behavior, where people draw on past experience when trying to solve new problems. This approach then does not require in-depth knowledge in a specific domain for solving the arising problems, and this is a main advantage of using CBR for effective results when there is only shallow knowledge of the domain. Case-based reasoning basically means using old experiences to understand and solve new problems, by reasoning from a previous situation resembling the current one [3].

An advantage of solving problems using CBR is described by [4]: "Neural network systems cannot provide explanations of their decisions and rule-based systems must explain their decisions by reference to their rules, which the user may not fully understand or accept. On the other hand, the results of CBR systems are based on actual prior cases that can be presented to the user to provide compelling support for the systems conclusions." Case-based reasoning builds on the idea that human expertise is not composed of formal structures like rules, but of experience: a human expert reasons

by relating a new problem to previous ones. An advantage of CBR over rule-based systems is that experience in CBR consists of examples of problems in a specific domain with their associated solutions that are readily available and can be easily acquired, on the other hand it is difficult to capture that knowledge in a set of rules [5]. A more detailed comparison between CBR and other methodologies will be given in the next chapter.

The following figure, Figure 1.1 from [6], illustrates the basic process of case-based reasoning and learning. The CBR cycle proposed by [7] consists of four sequential steps:

- *Retrieve*: one or more cases are selected from the case base, which have highest similarity to a query based on an underlying similarity measure.
- *Reuse*: the information and knowledge from the solutions of the retrieved cases, are used to solve the new problem.
- *Revise*: the selected solution is verified, corrected or improved.
- *Retain*: the new experience is incorporated into the existing case base for future problem-solving.

Being rooted in cognitive psychology [8], CBR is now considered as a proper subfield of Artificial Intelligence (AI). As a specific approach to knowledge engineering and knowledge-based systems design, it is closely related to machine learning, information retrieval, databases, semantic web, and knowledge management. Research in CBR has put much emphasis on combining methods from these areas to tackle specific problem tasks, such as diagnosis, planning, product recommendation, and experience management [9]. Moreover, much effort has been invested in implementing case-based systems for certain application domains, such as medicine [10] and the health sciences [11]. From a methodological point of view, CBR research has made significant and original contributions to similarity modeling, similarity-based retrieval, and adaptation.

Case-based reasoning has proved to be a successful problem-solving paradigm over the past years. Despite its great practical success, work on the theoretical foundations of CBR is still under way. In addition, a coherent and universally applicable methodological framework for case-based reasoning was yet missing. We also believe

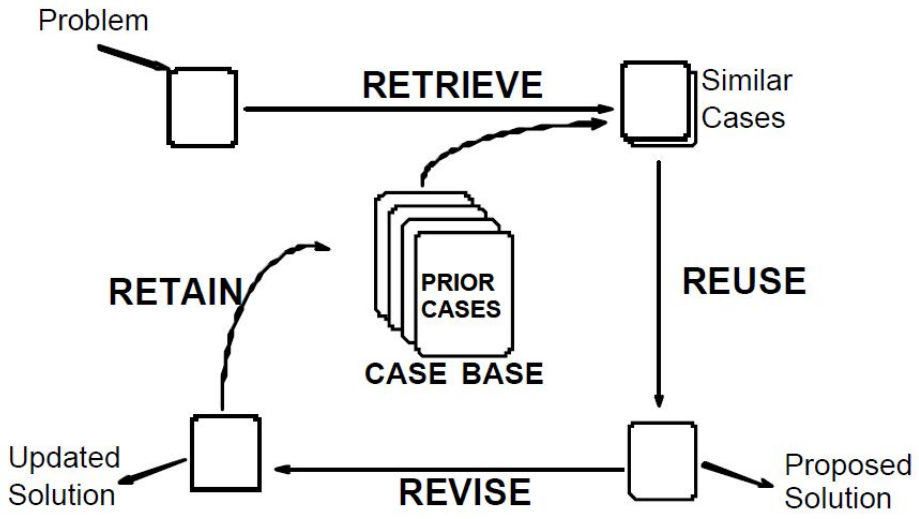


**Figure 1.1:** Case-based reasoning cycle

that although CBR systems exist and are used in many practical applications, there are some applications which might require solutions of a more flexible nature rather than a strict solution which is provided by standard CBR. These kind of applications exist in field areas which may have a very complex solution space as for example optimization problems. For those problems, specifying one optimal solution is almost impossible and in this case preference-relations of solutions (one solution preferred over another) would be a good remedy for the complex nature of solution representation in that field area. The aforementioned preference relations could be preferences of a user for example in qualitatively choosing one solution over another, such as choosing a preferred design or a preferred image or a preferred recipe of a meal over another. These preferences could be recommendations given by an expert in a specific domain, such as a doctor giving a preference for one treatment over another, or an architect giving a preference of some design fitting certain specifications more than another design, or a pharmacist recommending one drug over another for a person having some symptoms depending on his/her age and health status. In general, the preferences are qualitative choices based on some basis for comparison, where quantitative values, such as specifying for example

how good an image is or a tastiness of a meal, are difficult or impossible to specify. The lack of a methodological framework in CBR which is based on theoretical foundations, and the absence of a case-based reasoning framework incorporating a preference-based approach, inspired the strong motivation for our work.

A preference-based approach to case-based reasoning (CBR) has recently been advocated in [12]. The goal of preference-based CBR, or Pref-CBR for short, is to develop a coherent and universally applicable methodological framework for CBR on the basis of formal concepts and methods for knowledge representation and reasoning with preferences. Building on general ideas and concepts for preference handling in artificial intelligence (AI), which have already been applied successfully in other fields [13, 14, 15], the goal of this work is to establish a theoretically sound framework for the use of preferences in solving problems, using case-based reasoning methodology. The work in this thesis consists of the development of the above mentioned framework, the Pref-CBR problem-solving framework. To optimize this framework further, we consider the importance of the concept of similarity in CBR and we introduce methods for learning similarity measures. To ensure efficiency and maintenance of the whole CBR system, we also support our framework by developing case base maintenance strategies which specifically fit our Pref-CBR framework.

## 1.1 Preference-based case-based reasoning

As introduced in [12], a preference-based approach to CBR appears to be appealing for several reasons, notably because the case-based experiences lend themselves to representations in terms of preference relations quite naturally. In addition, the flexibility and expressiveness of a preference-based formalism well accommodate the uncertain and approximate nature of case-based problem solving. In this sense, the advantages of a preference-based problem solving paradigm in comparison to the classical (one optimal solution) one, which have already been observed for AI in general, seem to apply to CBR in particular. These advantages are nicely explained in [16]: "Early work in AI focused on the notion of a goal—an explicit target that must be achieved—and this paradigm is still dominant in AI problem solving. But as application domains become more complex and realistic, it is apparent that the dichotomic notion of a goal, while adequate for certain puzzles, is too crude in general. The problem is that in many

contemporary application domains [...] the user has little knowledge about the set of possible solutions or feasible items, and what she typically seeks is the best that's out there. But since the user does not know what is the best achievable plan or the best available document or product, she typically cannot characterize it or its properties specifically. As a result, she will end up either asking for an unachievable goal, getting no solution in response, or asking for too little, obtaining a solution that can be substantially improved."

A first step taken toward preference-based CBR was made in [12], by addressing the important part of case-based inference, which is responsible for predicting a "contextualized" preference relation on the solution space. More specifically, the latter consists of inferring preferences for candidate solutions in the context of a new problem, given knowledge about such preferences in similar situations. The search-based problem-solving framework developed in this thesis embeds this inference procedure. More specifically, case-based problem solving is formalized as a search process, in which a solution space is traversed through the application of adaptation operators, and the choice of these operators is guided by case-based preferences.

## 1.2 Similarity measures

The main idea in CBR is solving new cases by reusing the knowledge from previously solved cases, which are stored in a case base. To this end, it is important to choose the *useful* cases for the current problem-solving episode, from all the cases in the case base. The CBR system chooses these useful cases based on some *similarity measures*, which are difficult for the users to express and yet are based on some heuristics (rule-of-thumb or argument) of the knowledge incorporated in a specific application domain.

It is clear that learning the similarity measures improves the choice of the useful cases to use for solving a new problem, leading to finding a better solution for the current problem, thus improving the overall performance of a CBR system. Since the concept of similarity lies at the heart of CBR, the success of a CBR system strongly relies on the specification of a suitable similarity measure [17]. As domain knowledge provided by human experts is mostly not sufficient to manually provide an optimal measure, machine learning algorithms could be used to learn these similarity measures efficiently.

In standard CBR, it is common to define similarity measures on the problems. In Pref-CBR, since the preference relations are on solutions, it is hence important to also define a similarity measure on the solutions and not only on the problems. As a consequence, the performance and effectiveness of Pref-CBR is strongly influenced by the distance measure (smaller distance means higher similarity) between the solutions: the better this measure captures the true differences between solutions, the more effective Pref-CBR will be. Thus, the idea is to make use of the experience collected in a problem solving episode, not only to extend the case base through memorization of preferences, but also to adapt the distance measure between the solutions. For completeness, we also integrate a distance learning module to learn the distance measure between the problems.

## 1.3 Case base maintenance

As the case base contains all the solved problems along with their corresponding solutions, a case base can quickly expand in size and eventually its efficiency is hampered. In Pref-CBR, a case is not only stored in the case base along with one corresponding solution, but rather with a case along with its corresponding preferences over solutions. It is clear that simply storing each encountered problem along with a set of associated preferences is not advisable, especially since a case base of that type may quickly become too large and hamper efficient case retrieval; besides, many of the preferences collected in a problem solving episode will be redundant to some extent. In CBR, this problem of redundancy has been addressed by methods for *case base maintenance* [18, 19]. Such methods seek to maintain the problem solving competence of a case base thanks to *case base editing* strategies, including the removal of misleading (noisy) or redundant cases. Case base maintenance (CBM) proved essential to guarantee the efficiency and performance of CBR systems. According to the aforesaid about a case base growing rapidly in Pref-CBR, CBM might be even more critical for preference-based than for conventional CBR.

Due to the above mentioned reasons for the importance of maintaining a case base, we address the problem of case base maintenance in Pref-CBR and develop a method for applying it specifically in our framework. We develop some CBM strategies that can be integrated into our Pref-CBR framework to increase the efficiency of the case base

and yet maintain its performance. Despite being inspired by existing CBM techniques for conventional CBR, our strategy is specifically tailored to our framework and exploits properties of the underlying preference-based representation of problem solving experience.

The thesis is organized as follows, the second chapter introduces the methodology of CBR. The third chapter describes the developed preference-based CBR framework. In the fourth chapter, learning similarity measures is explained, and the case base maintenance method which is specifically designed for the Pref-CBR framework is discussed in Chapter 5. In Chapter 6 methodologies related to Pref-CBR are reviewed; these include different search strategies, as well as machine learning approaches. Chapter 7 describes the case studies, which illustrate the performance of the Pref-CBR framework and show the effect of learning similarity measures and applying the maintenance methods. Chapter 8 includes the summary, conclusion and suggested future work.

# 2

# Case-Based Reasoning

Case-based reasoning in general is the process of solving new problems based on previously solved problems. Humans are using case-based reasoning most of the time in their every day life. Doctors use case-based reasoning when they get a patient suffering from certain symptoms, and immediately think of previous patients having the same or similar symptoms, they use that previous experience to come up with an appropriate treatment for the new patient. This treatment could be taken exactly as it was given to a previous patient or the doctor might see a need for changing it a bit, or adapting it to the new patient. An auto mechanic who fixes a car engine by recalling a previous car that exhibited similar symptoms is using case-based reasoning. The same reasoning is used by a lawyer faced with a new case, an electrician repairing some power outages, a cook having some ingredients to cook a meal. All these people use case-based reasoning; they think of a similar case in the past and from the knowledge of this past experience they solve the current case.

CBR is a problem solving paradigm which is characterized by two major distinctions from other AI approaches; it is able to utilize specific knowledge from previous experience of concrete cases and it is considered as an incremental and sustained (continuous) learning approach [7]. Thus, a new problem is solved by adapting the solution of a similar case, rather than solving the case from scratch. Reuse of prior solutions helps increase problem solving efficiency by building on prior reasoning, rather than repeating prior effort [20]. Accordingly, a CBR system requires efficient techniques for retrieving cases (which are maximally similar to the problem), organizing and maintaining the case base, and adapting the stored cases to the problem at hand [21].

In this chapter, we will explain the problem-solving process of CBR and how CBR systems are used. We will discuss the relationships of CBR to other approaches. To better understand CBR, we will mention some practical applications which use CBR and will show why CBR systems are gaining interest and showing success.

## 2.1 Problem-solving CBR

CBR is considered as a psychological theory of human cognition and addresses issues in memory, learning, planning and problem solving [22]. In its simplest form, when a CBR system is faced with a new problem having a specific problem description, the system searches for other problems with similar problem descriptions. The solutions of those problems are used as the starting point for generating a solution for the new problem. Following the CBR notion of: similar problems have similar solutions, a solution for the new problem is found and maybe adapted if necessary [20]. The process of getting a new problem, finding similar problems and taking their solutions to adapt and find a final solution to that new problem, is called a problem solving episode.

It is important first to imagine what a case base looks like, what is the structure of the collected cases in the case base and how is the content of the cases organized in the case base? For ensuring effective retrieval and reuse of the cases from a case base, the description of the contents of the cases should be structured in such a way as to provide proper indexing of cases [7]. A case base can have a simple flat structure, where problems are represented by feature vectors and solutions are also represented by feature vectors or classes. As described by the authors in [7], a case base can also have a hierarchical structure, in which cases sharing similar properties are grouped together and the case memory looks like a network structure of categories, semantic relations, cases, and index pointers. A case base can also be structured as a dynamic memory model, proposed by [23], where similar cases are grouped together in clusters which are characterized by specific indexes. Properly structuring the case memory provides the ground basis on which the retrieval task will later efficiently operate on. The retrieval task will start with a problem description, a matching and selection of similar cases, based on the way the cases are organized and structured in the case base and finally providing best found matches [7].

CBR systems mostly exploit the nearest neighbor algorithm (NN) in their retrieval phase for finding similar cases to a new problem [24], where the *nearest* relation is computed using a similarity metric on the problem space. Given a description of a problem, a retrieval algorithm, using the indices in the case memory, should retrieve the most similar cases to the current problem or situation [25]. Commonly, CBR systems use mostly Hamming distance for finding nearest neighbors for an attribute-value representation of a problem, while for n-dimensional real vectors, Euclidean or Manhattan distance are often used [26, 27]. For other complex problems such as problems containing heavily inter-connected events (scheduling or time-tabling), the structured case representations require other similarity measures that incorporate deeper knowledge about how much modification of a case is required to fit a new problem [28]. An example of structured case representations are graphs, trees and semantic networks. The authors in [28] state that similarity based on feature-value representation is not sufficient in finding similarity between structured cases because the feature-value representation alone in this case is not enough to find the correspondence between the features in cases and characteristics of the solutions; similarity measures accordingly should consider geometric relationships, hierarchically-structured and model-based similarities. Some similarity assessment methods for structured cases include network-based retrieval methods, graph editing operations and sub-graph matching [6].

After retrieval of similar cases to a problem by using some similarity measures, there might be the need for adaptation of a similar case to find a solution to fit the new problem.

Shown in Figure 2.1 is an example of a problem-solving episode for problem x: first the nearest neighbors of problem x are retrieved using the NN algorithm, then a solution is adapted from their corresponding solutions to fit problem x and final solution y is obtained. All problems and their corresponding solutions are stored in the case base for future retrieval when needed. There are several adaptation methods which include: transformation adaptation (structure of solution is changed), substitution adaptation (some parts of solutions are replaced) and generative adaptaion (solution is recomputed and derived to fit new problem) [6]. In addition, the authors in [29] propose compositional adaptation, which in the case of existence of several similar cases to a new case, their corresponding solutions would be combined in some efficient way to obtain a final solution. As will be described in the next chapter, our Pref-CBR framework

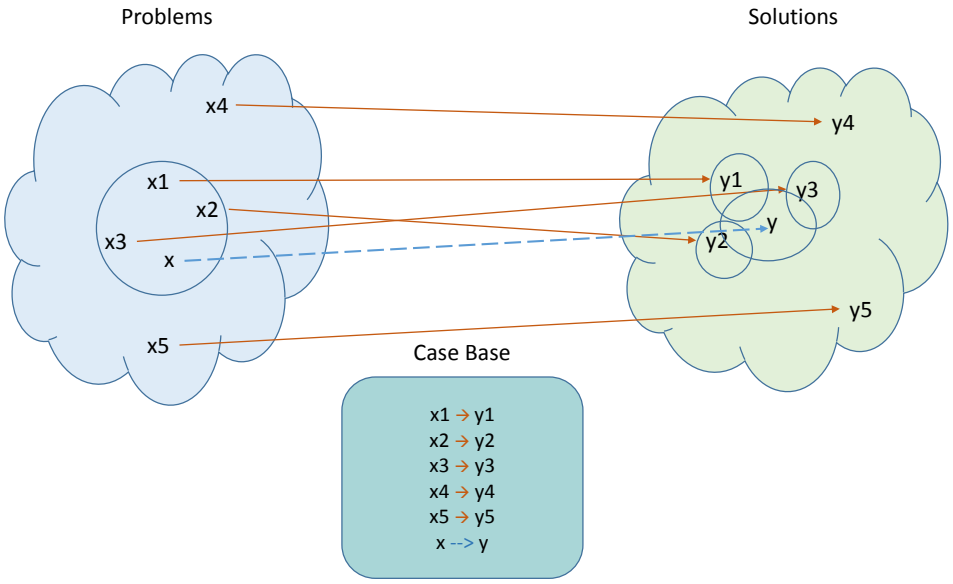


**Figure 2.1:** An example of problem-solving in CBR

incorporates in a way this idea of combining the solutions of some similar cases to yield a final solution.

Figure 2.2 from [22] illustrates the CBR cycle as a flowchart, where boxes represent processes and ovals represent knowledge structures:

Let us assume that we have as a new problem some given resources and measurements of a house, and the solution should be a suitable design which optimally makes use of the given resources and the area of the house.

- *Assign Indexes*: features of a new problem are assigned as indexes characterizing that problem. For example specifications of number of rooms of a house and area as well as resources, and the problem is to find a suitable design.

- *Retrieve*: depending on the structure of the cases, some suitable similarity measures are used. Indexes are used to retrieve a past similar case containing a prior solution, in our example we would retrieve a previous design.

- *Modify*: some adaptation method is applied on the retrieved solution to fit the new problem. Old solution is modified to fit the new situation, resulting in a

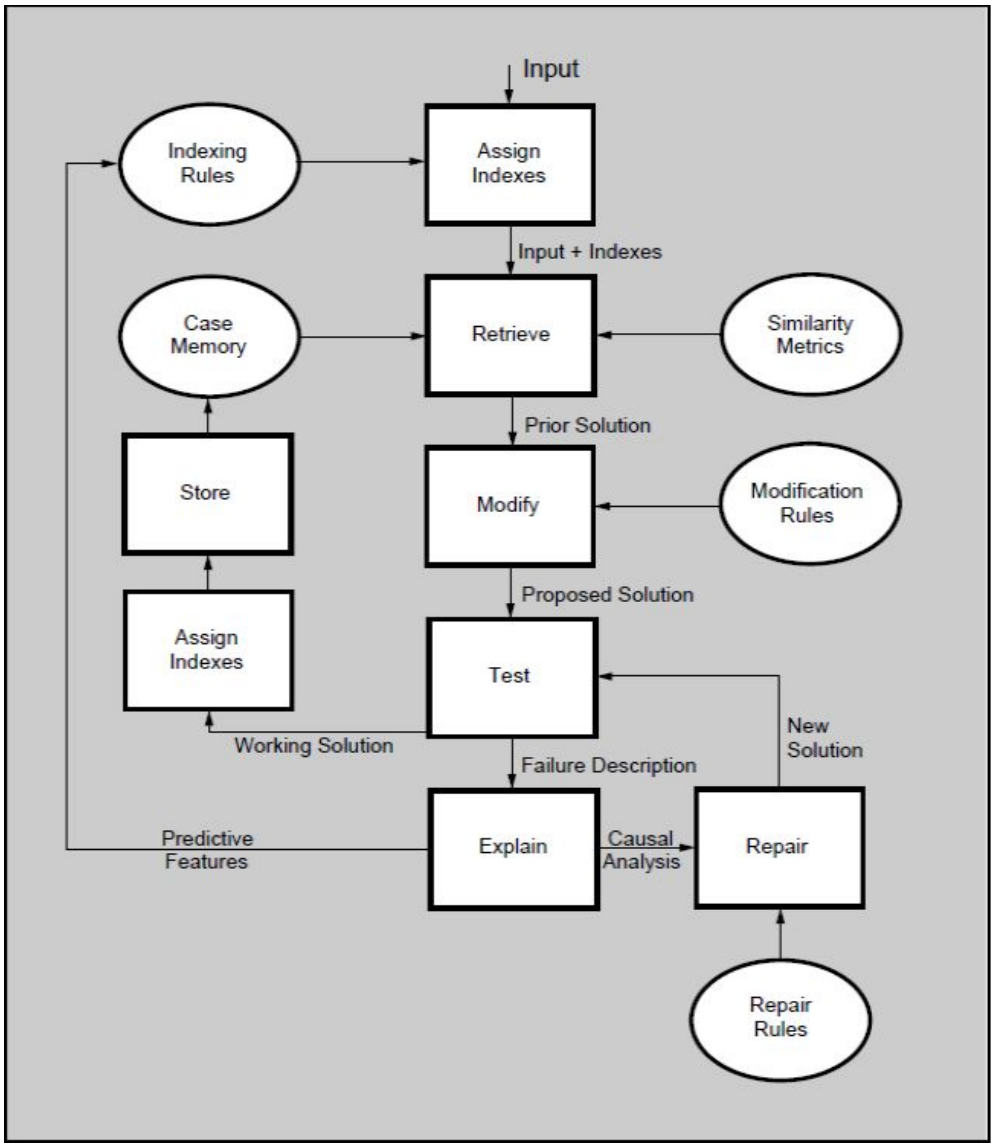


**Figure 2.2:** Case-based reasoning flow chart

proposed solution. The design is modified to fit requirements of the features of the problem.

- *Test*: solution is tried out to see whether it succeeds or fails. Does the design include all required resources and user requirements as well as using the specified area optimally? Check the prototype design.
- *Assign and Store*: if solution is successful then assign indexes and store a working solution in the case memory. Store the successful design in the case base.
- *Explain, Repair and Test*: if solution fails then explain the failure, repair the working solution and test again. Failed plan is repaired and revised solution is tested. In our example we can state for example that for a specific area of a house, it is not recommended to have more than 4 rooms because of electricity and power supply resources.

Supporting the above processes are the following knowledge structures also explained by [22]:

- *Indexing Rules*: identify predictive features in the input.
- *Case Memory*: database of experience.
- *Similarity Metrics*: are used to decide which case is more like the current situation.
- *Modification Rules*: knowledge about what factors could be changed and how to change them.
- *Repair Rules*: rules for what kind of changes are permissible.

By design, CBR systems do not need deep general knowledge and can be used with just some initial knowledge and further knowledge could be manually added or learned over time [30]. Problem-solving CBR involves situation assessment, case retrieval and similarity assessment/retrieval. In addition to these tasks, the similarities and differences between new and previously solved cases are used to determine how the solution of the previously solved cases can be adapted to fit the new situation [20]. As we learned up to this point, problem-solving systems will store and adapt prior solutions.

The notion of a quality of a solution to a certain problem has been introduced by [31], where quality represents some kind of utility or a degree of correctness. In the latter, the authors propose a formal model of transformational adaptation which defines a quality function that assigns a quality value to each problem-solution-pair; solutions with a higher quality are preferred over solutions with a lower quality value. An alternative approach mentioned in [20] is to store and reuse traces of how those solutions were derived, instead of storing only the actual solutions. In our developed Pref-CBR framework, which will be explained in detail in the next chapter, we implement the aforementioned ideas; we generate preferences over solutions and we do not only store solutions, but we store the traces of how those solutions were derived.

## 2.2 Conversational CBR

Conversational CBR (CCBR) is a form of interactive case-based reasoning systems, it is a means of providing more effective support for interactive problem solving. CCBR is used in applications where the user does not necessarily have much domain knowledge, thus helping the user to reach a goal or find a solution for a problem, or help a user to make a decision. In this section we briefly describe how CCBR systems work and some applications which use these systems. We also show the resemblance of CCBR to our Pref-CBR framework.

### 2.2.1 CCBR systems

Most commercial CBR systems use an interactive questioning process, for example to assist maintenance personnel with fault diagnosis tasks or in e-commerce to help a user to buy certain products or in medical decision making [32, 33, 34, 35]. CCBR systems require a user to initially input a brief free-text description of his/her problem and accordingly the system then constructs a problem specification by interactive problem assessment [33]. As [33] describe the CCBR systems, they show the advantage to progressively rank and display the top matching cases' solutions, not requiring the user to have a prior knowledge about these relevant cases. It can be stated that CCBR systems can be characterized as interactive systems that, via a mixed-initiative dialogue, guide users through a question-answering sequence in a case retrieval context [32].

A popular use for CCBR systems is for product recommendation. In this context, the query represents the preferences of the user according to the attributes of the available products [36]. Accordingly, at each stage of the recommendation dialogue, the system selects the most useful attribute (feature) and asks the user for a preferred value of this attribute and the product which is most similar to the query is retrieved and showed to the user. As mentioned by [36], the dialogue terminates based on a set *termination criterion*, or until no further attributes remain. The termination of the problem solving is unsuccessful when the system cannot find a good match or when there are no further relevant questions to ask [33].

CCBR systems can also be used in customer support domains by playing the role of customer support agents. They have been successfully used to improve knowledge management in corporate activities by aiding in problem solving, and they have also been used in designing the user-interface of e-commerce websites [34]. Another successful application where CCBR systems have been used is in medical classification and diagnosis, interactive fault diagnosis and helpdesk support [35]. It is worth mentioning that in contrast to traditional CBR approaches, it is not assumed that the problem to be solved has an available description. A problem description is instead elicited by the system, aiming to minimize the number of questions that the user is asked before a solution is reached [35].

An important issue addressed in CCBR is the attribute selection strategies that aim to minimize the length of the interactive dialogue with the user. Features are selected by maximizing information gain and avoiding redundant questions in conversation with the user. For example from the reply of the user to certain queries, some features of the problem are refined according to the user's replies to better find similar cases fitting his/her needs. This in turn increases conversational efficiency, thus reducing the length of the dialogue [34]. Some potential benefits of minimizing the number of questions being presented to the user include avoiding frustration of users and reducing network traffic in e-commerce applications [36]. The criteria for termination of problem-solving dialogues is a very challenging issue for balancing the trade-offs between solution accuracy, and problem-solving efficiency [35, 36].

It can then be plausible to say that the conversational CBR systems are closely related to our Pref-CBR framework, in the sense that there is some form of feedback given during the search process, and based on this feedback the chosen solution for the

query is found. In our framework, this feedback is given by what we call an *oracle*, where this oracle does not necessarily represent a user but can also represent a program, an expert or a complex database and is described in more detail in the next chapter. In CCBR the problem can initially be defined by the user and it is more restricted to operate in certain environments and specific applications, but in our framework the interference of a user is only a part of the framework (generating preferences) which improves the initial solution to obtain the best possible solution based on the user's preferences. In addition, in our framework the query does not necessarily represent the user's preferences, but the user's preferences are rather generated within the search to adapt the solution chosen based on solutions of similar cases to the query. Our framework is more generic and can be used in many applications in different fields, it is not only restricted for applications including user interaction. For some applications having structured problem/solution spaces, as for example providing the best graph or an optimal tour, our framework can be used while CCBR systems would not be suitable for such applications.

### 2.2.2 Recommender systems

Recommender systems have become popular over the past years and are applied in a variety of applications. The most popular and well known applications are probably movies, music, news, books, research articles, social tags, search queries and products in general. However, there are also recommender systems for experts, jokes, restaurants as well as financial services. It should be noted that recommender systems are not part of CBR systems, but case-based recommenders are special recommender systems. The success of case-based recommenders depends on two important domain properties: the items need to be described by well-defined features, and users must have some understanding of these features and how they are related to their requirements [37]. The details of a target problem are elicited from the user, often with questions selected or ranked in order of usefulness by the system [35].

Many recommender systems seek feedback from the user as part of the recommendation process, and based on this feedback these systems perform information filtering and user profiling using machine learning techniques and adaptive user interfaces, to help users find the information they are seeking [38]. A key feature which separates recommender systems from more conventional information retrieval technologies such

as search engines, is their conversational character. The conversational recommender systems guide users through a product space for example, and by eliciting user feedback, alternatively making product suggestions, which prioritize products that best satisfy the user's preferences [39].

The recommender systems incorporate four different feedback strategies from the user, which have been explained by [38]. Value elicitation (providing a specific value for a specific feature), tweaking (a directional preference for a particular feature), ratings-based (rate-recommended cases according to their suitability) and preference-based (selecting one of current set of recommendations that is closest to their requirements).

The last form of feedback (preference-based), is the lowest cost form of feedback as it only requires a user to express a simple positive or negative preference for one of the recommended cases. This resembles closely our preference-based strategy which is incorporated in our Pref-CBR problem solving framework. The recommender systems which use the preference-based feedback have the closest resemblance to our framework, where the qualitative preferences play the major role in the search process for a suitable solution for a problem. Recommender systems though require well-defined features for queries, which the user understands to be able to relate them to his or her requirements, which is not the case in our Pref-CBR framework. The preferences of the user are used to optimize a solution for a query, where user only gives preferences of two suggested solutions, not necessarily having to understand or know the features of the problem. Our Pref-CBR framework can be used for *searching* for a suitable solution and not just merely for *analyazing* to find a suitable solution.

## 2.3 Relationships of CBR to other approaches

The following approaches are in some way connected to CBR and accordingly play some role within CBR. The relationships of CBR to other approaches is illustrated in the following diagram in Figure 2.3 by [40]:

- *Database Management Systems*: the connection between database management systems and CBR systems is that both contain data elements of interest to be retrieved if needed [40]. In contrast to database queries which extract only items which match exactly certain retrieval criteria mentioned in the query, CBR queries rank items retrieved using similarity. Unlike CBR systems which give solutions

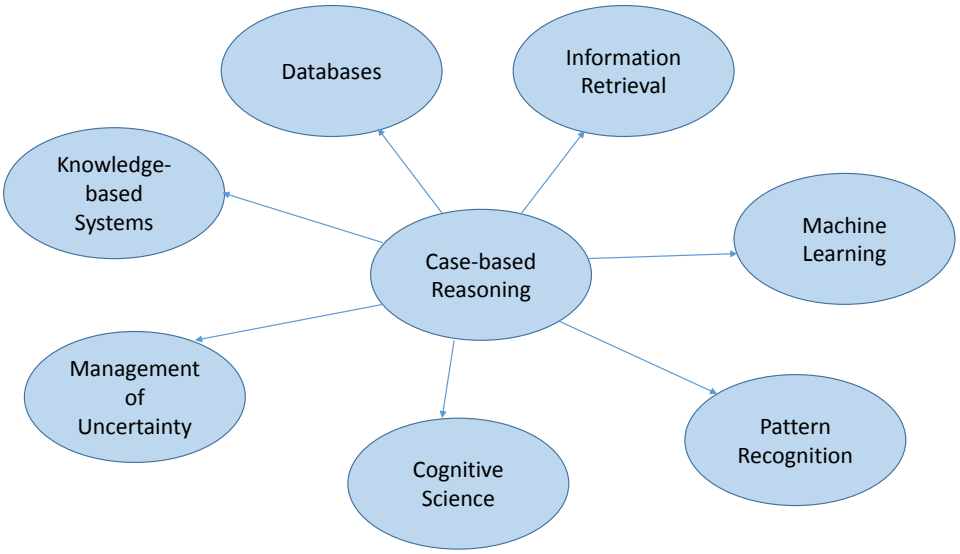


**Figure 2.3:** Relations of CBR to other areas

that could be adapted to a specific query, databases do not consider reuse or adaptation and certainly do not have the notion of a problem and solution that exists in CBR. In general, the relationship between CBR and database management systems can be described as follows: a case base can obviously be seen as a kind of database which needs an organization scheme, indexing of cases for retrieval, and storage of cases for later reuse.

- *Information Retrieval Systems*: information retrieval systems are similar to CBR systems in the sense that both try to retrieve best possible answers to queries. Information retrieval systems translate the needs of information of a user to a query, which is compared to descriptions of information found in accessible documents [41]. The difference between information retrieval systems and CBR systems is that the former retrieve only documents and can then be more closely related to textual CBR, where both refer to written text [40]. It can then be concluded that information retrieval systems represent a form of CBR systems which specifically handles cases including text.

- *Knowledge-Based Systems*: the difference between knowledge-based systems and CBR systems is that the former focus on retrieval of implicitly stored knowledge

using a logical inference process (an inference engine represented mostly by if-then rules), while the latter retrieve explicitly stored knowledge [40]. Building knowledge-based systems requires expert knowledge while no expert knowledge is needed when building CBR systems. As a consequence, former systems give valid answers but updating the knowledge base is problematic, while the latter systems give approximate answers but are easier to maintain and are more systematic. We can also imagine that a case base is basically a knowledge base; it contains stored cases containing knowledge (problems, solutions and adaptation knowledge which is the experience used for solving new cases). CBR can be considered as a tool which technically supports knowledge-based systems, by organizing, restructuring and memorizing the knowledge to be managed [42]. As highlighted by [43], we can view CBR as a means of providing methodological support to a knowledge management system's activities for a complete knowledge sharing experience within a learning environment.

- *Machine Learning*: as [40] simply states, machine learning systems and CBR systems both operate on presented examples that we call experiences. A common aim for both systems is to reduce knowledge acquisition, unlike knowledge-based systems. Similar to rule-induction algorithms of machine learning, CBR starts with a set of cases or training examples to form generalizations of these examples by identifying the commonalities between a retrieved case and the target problem. In CBR, generalizations are made based on the given target problem (lazy learning), while in other models such as rule-induction algorithms generalizations are drawn from training examples before the target problem is known (eager learning) [44]. It can be noted that in machine learning, CBR can be seen as a specific type of learning, mainly instance-based learning. Instance-based (lazy learning) methods in machine learning include case-based reasoning methods that use more complex, symbolic representations for instances [45]. We can also consider that within CBR, some machine learning methods could be used for different purposes such as in the search process and for learning similarity measures.

- *Cognitive Science*: cognitive science is the interdisciplinary scientific study of the mind and its processes and includes research on intelligence and behavior, especially focusing on how information is represented, processed, and transformed

[46]. Some of the basic roots of CBR are located in cognitive science and have a relation to cognitive psychology; solving complex problems similar to the way humans understand and memorize experiences with the integration of computational techniques [40].

- *Analogical Reasoning*: as [20] pointed out: "CBR can be viewed as fundamentally analogical; the CBR process emphasizes the performance of analogical reasoning and the feedback of evaluation, in order for a case-based reasoner to learn its lessons while adding a new experiential episode of success or failure to its memory. Analogical reasoning has many overlaps with CBR and both have in common the way in which they make use of past experiences". The difference between analogy and CBR is stated by [40], analogy operates on a more strategic level than CBR, where the former is based on comparing abstract structures of two entities, while the latter relies on the concept of similarity by focusing on the attributes shared by the two entities. Analogy aims at learning and generalizing by examining at least four entities, while it can be sufficient in CBR to solve a new problem from one similar case.

- *Management of Uncertainty*: uncertainty in CBR can occur due to information which maybe missing, or when two different descriptions are describing the same problem causing uncertainty of which description to be used, or when there is uncertainty of having the correct solution [47]. The need for suitable methods to model and reason under uncertainty has led to an increased interest in formalizing parts of the CBR methodology, by combining CBR with methods of uncertain and approximate reasoning and soft computing [48]. It is important to manage uncertainty in CBR as it is used in many applications for decision making. An example of using CBR for decision making is in enterprises, since it is considered as a suitable decision-making paradigm because it resembles closely the way managers make decisions [49]. A helpful description of decision making, uncertainty and CBR is given by [50]: "the idea of case-based decision making has recently emerged as a new paradigm for decision making under uncertainty. It combines principles from decision theory and case-based reasoning".

## 2.4 Practical applications in CBR

CBR systems have been used for many tasks in classification, interpretation, scheduling, planning, design, diagnosis and explanation [20]. An approach for using CBR in story plot generation has been proposed by [51], where a CBR process is used to generate plots from a user query specifying an initial setting for the story by using an ontology to measure the semantic distance between words and structures in a text. CBR has been used for aiding designers to recall a relevant design and reuse it to help generate a solution to a new design [52]. The authors in [53] propose to use case-based reasoning to assist architects. As stated by [53], people generally find it easy to use analogs in reasoning but find it difficult to remember the right ones, thus CBR systems can be well used as human-aid systems to help people do better analogical reasoning.

CBR can also be used in interactive aiding of decision making, which is beneficial especially for novices who lack a sufficiently complete collection of prior experiences [54]. Such an example of a decision support system can be found in help-desk systems (first level support) in companies, where they can be used by customers who have problems with products or services from that company [55]. More recently, CBR techniques have also been applied in the design and construction of simulation-based learning systems and educational games; the iterative cycle of applying knowledge, interpreting feedback, explaining results and revising memory provides a model for active promotion of learning [56].

A case-based approach to knowledge navigation has been proposed by [57]. In their work, the authors propose three agents: browsing systems (Find-Me Systems), preference-based task organizers (BUTLERS) and internet news group agents (CORRESPONDENTS). All three types of systems help users navigate through information spaces, and either find or construct responses to fit the needs of those users. CARE-PARTNER is another knowledge support system on the world wide web which integrates CBR; it is a complex medical application running on the internet for physicians to follow up post-transplant patient care. The system integrates rule-based reasoning, information retrieval and case-based reasoning, where CBR is used for refining and completing the knowledge of the system by learning from experience and improving results over time [58].

For illustrating how a CBR system works, let us take the CHEF system (builds new recipes on the basis of requests of a user from previously stored recipes) as a running example [59]. Recalling Figure 2.2 from [22] which illustrates the CBR cycle as a flowchart, the following procedure describes an example from the cooking domain illustrating how the CHEF system works:

- *Assign Indexes*: generate a beef and broccoli recipe.
- *Retrieve*: retrieve a stored beef and green beans recipe.
- *Modify*: apply modification rule to substitute green beans with broccoli and edit cooking time.
- *Test*: plan is executed in simulator and adapted recipe goals are checked against expectations.
- *Assign and Store*: broccoli is soggy!
- *Explain, Repair, Test*: beef releases water and broccoli steams. Add broccoli later on in the dish. Test for crispness of broccoli.

## 2.5 Conclusion

As we have seen, CBR systems can be used in different domains and have shown success in many practical applications. The analogy-based method which is used in CBR, recalling instances from a case base which are similar to a new given case, has been the reason for the support of these systems to humans. The reason for this is that they model the human behavior with the computational capability to support humans in knowledge extraction, diagnosis, problem solving and decision making. In this chapter we introduced key principles of CBR, its basic problem-solving process, its relationships to other approaches and some practical applications of CBR systems. In the next chapter we will highlight the difference between preference-based CBR and standard CBR and we will describe in detail our Pref-CBR framework.

# 3

# Preference-Based Case-Based Reasoning

The common knowledge representation in a case base, that we are generally familiar with, is a case and its solution. A major distinction of our Pref-CBR framework from the standard CBR systems is our knowledge representation in the case base, which is a case and its set of preferences over solutions. Our framework incorporates a search mechanism which operates on solving new problems using solutions of previously solved similar problems, where the solutions in our case are preferences over solutions rather than single solutions. This preference-based approach for problem-solving can be logical to use in certain domains, where preferences need to be included during the search process or when there is no one distinct solution for a problem. Some examples of such domains are the medical domain (two therapies could be used for curing a patient but one is preferred over another for that specific patient), cooking domain (two meals could be offered by using specific ingredients but one is preferred over another depending on user's preferences), traveling domain (two trips could be offered satisfying requirements of user but one is preferred over another depending on user's choice of priorities whether it be price or luxury, etc.).

There are also other domains which incorporate a more complex structure, where our Pref-CBR problem-solving framework has an important advantage over standard CBR as well as over other standard machine learning methods. The structure of the problem/solution spaces can be simple, where problems and solutions are represented by feature vectors and the spaces are continuous where similarity between problems or

between solutions can be distinct and highly expressive. For these simple spaces, standard CBR as well as many machine learning methods could be efficiently used. There are problem/solution spaces though which are more complex, such as having geometric representation or discrete spaces such as procedures, item sets or permutations in which similarity between problems or solutions are more difficult to assess and distinguish. For those complex spaces our Pref-CBR framework can show an advantage over standard CBR or other machine learning methods, since the comparisons for nearest neighbor search and pairwise preferences rely on distance measures which allows for spaces to have very unique structures. Our Pref-CBR framework can efficiently handle such complex spaces, only requiring the definition of a similarity measure in the space to implement the search. To be able to implement a good performance of the search it is essential to define distance values that are expressive enough to properly distinguish between different problem instances and to also properly distinguish between different solutions. Contrary to other methods, which are typically operating on feature representations, our framework is generic and can operate in different domains, even ones having complex structures. In Chapter 7 we describe two applications, one including item sets and the other including the traveling salesman problem, where we show how our framework operates within complex solution spaces provided that a there is a good definition of expressing similarities (distances) between solutions.

The preferences over solutions are generated by what we call an *oracle*. An oracle in our framework represents an expert of some kind: a doctor, a pharmacist, a cook, a designer, a program or a large and complex database. The idea of the *oracle* is that it gives a minimum amount, yet very knowledgeable and useful information, for finding an optimal solution to a given problem, by generating preferences over solutions. Needless to say, queries to the oracle are expensive and accordingly we try to keep this to be minimal. The queries to the oracle, in addition to our inference method for problem solving, lead us to find an optimal solution for our problem at hand.

The flexibility and expressiveness of a preference-based formalism well accomodate the uncertain and approximate nature of case-based reasoning, making a preference-based approach to CBR seem very appealing [12]. The preference-based approach for solving problems has been used successfully by several applications. Preference-based navigation can be used for online information access, for electronic catalogs and in general for e-commerce applications [60]. E-commerce recommender systems help

consumers to locate products within a complex product space by using a technique of adaptive selection which employs preference-based feedback [61]. Conversational recommender systems often elicit feedback from the user during the recommendation process and this feedback is used to elaborate the query of the user and thus guide the next recommendation cycle [62]. A preference-based approach in CBR can also be used in decision-aiding, where a decision maker should be involved to express his/her preference to finally reach a decision [63].

Due to the attractiveness of using a preference-based approach in some domains for problem-solving, as well as accomodating the uncertain nature of CBR, we got the motivation to create a generic Pref-CBR problem-solving framework which can be used in many simple or even complex domains efficiently. In this chapter, the difference between conventional CBR and preference-based CBR is highlighted and the Pref-CBR problem-solving framework is thoroughly explained and its benefits will be shown. The main idea of Pref-CBR will be discussed, the case-based inference will be explained as well as a description of how this inference is then embedded in a more general search-based problem solving framework.

## 3.1 Conventional CBR vs preference-based CBR

Experience in CBR is most commonly (though not exclusively) represented in the form of problem/solution tuples $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{X} \times \mathbb{Y}$, where $\boldsymbol{x}$ is an element from a problem space $\mathbb{X}$, and $\boldsymbol{y}$ an element from a solution space $\mathbb{Y}$. Despite its generality and expressiveness, this representation exhibits some limitations, both from a knowledge acquisition and reuse point of view.

Preference-based CBR replaces experiences of the form "solution $\boldsymbol{y}$ (optimally) solves problem $\boldsymbol{x}$" by weaker information of the form "$\boldsymbol{y}$ is better (more preferred) than $\boldsymbol{z}$ as a solution for $\boldsymbol{x}$", that is, by a preference between two solutions *contextualized* by a problem $\boldsymbol{x}$. More specifically, the basic "chunk of information" we consider is symbolized in the form $\boldsymbol{y} \succeq_{\boldsymbol{x}} \boldsymbol{z}$ and suggests that, for the problem $\boldsymbol{x}$, the solution $\boldsymbol{y}$ is supposedly at least as good as $\boldsymbol{z}$.

This type of knowledge representation overcomes several problems of more common approaches to CBR. In particular, the representation of experience is less demanding: As soon as two candidate solutions $\boldsymbol{y}$ and $\boldsymbol{z}$ have been tried as solutions for a problem

**Table 3.1:** Notations

| notation | meaning |
|---|---|
| $\mathbb{X}$, $\boldsymbol{x}$ | problem space, problem |
| $\mathbb{Y}$, $\boldsymbol{y}$ | solution space, solution |
| **CB** | case base (storing problems with preferences on solutions) |
| $S_X$, $\Delta_X$ | similarity/distance measure on $\mathbb{X}$ |
| $S_Y$, $\Delta_Y$ | similarity/distance measure on $\mathbb{Y}$ |
| $\mathcal{N}(\boldsymbol{y})$ | neighborhood of a solution $\boldsymbol{y}$ |
| $\boldsymbol{y} \succeq_{\boldsymbol{x}} \boldsymbol{z}$ | $\boldsymbol{y}$ is better (more preferred) than $\boldsymbol{z}$ as a solution for $\boldsymbol{x}$ |
| $\mathfrak{P}(\mathbb{Y})$ | class of preference structures on $\mathbb{Y}$ |
| $\mathcal{P}(\boldsymbol{x})$ | set of (pairwise) preferences associated with a problem |
| CBI | case-based inference using ML estimation (see equation (3.5)) |

$\boldsymbol{x}$, these two alternatives can be compared and, correspondingly, a strict preference in favor of one of them (or an indifference) can be expressed. To this end, it is neither required that one of these solutions is optimal, nor that their suitability is quantified in terms of a numerical utility.

Conventional CBR systems, as stated in [54], have been mainly designed as automated problem solvers for producing a solution to a given problem by adapting the solution to a similarly solved problem. Such systems may have limited success in real-world applications for the following reasons:

- *Existence of correct solutions*: It assumes the existence of a "correct" solution for each problem, and implicitly even its uniqueness. This assumption is often not tenable. In the cooking domain, for example, there is definitely not a single "correct" recipe for a vegetarian pasta meal. Instead, there will be many possible alternatives, maybe more or less preferred by the user.

- *Verification of optimality*: Even if the existence of a single correct solution for each problem could be assured, it will generally be impossible to verify the optimality of the solution that has been produced by a CBR system. However, storing and later on reusing a suboptimal solution $\boldsymbol{y}$ as if it were optimal for a problem $\boldsymbol{x}$ can be misleading. This problem is less critical, though does not dissolve, if only "acceptable" instead of optimal solutions are required.

- *Loss of information*: Storing only a single solution $\boldsymbol{y}$ for a problem $\boldsymbol{x}$, even if it can be guaranteed to be optimal, may come along with a potential loss of information. In fact, during a problem solving episode, one typically tries or at least compares several candidate solutions, and even if these solutions are suboptimal, preferences between them may provide useful information.

- *Limited guidance*: From a reuse point of view, a retrieved case $(\boldsymbol{x}, \boldsymbol{y})$ only suggests a single solution, namely $\boldsymbol{y}$, for a query problem $\boldsymbol{x}_0$. Thus, it does not imply a possible course of action in the case where the suggestion fails: If $\boldsymbol{y}$ is not a good point of departure, for example since it cannot be adapted to solve $\boldsymbol{x}_0$, there is no concrete recommendation on how to continue.

Due to these limitations, there has been a search for new paradigms for the utility of CBR systems for decision support [54]. Making further steps in avoiding such limitations, we take a first step toward developing a preference-based CBR framework. In our search-based problem solving framework we embed a case-based inference component, which is responsible for predicting a "contextualized" preference relation on the solution space [12]. More specifically, it consists of inferring preferences for candidate solutions in the context of a new problem, given knowledge about such preferences in similar situations. In our Pref-CBR framework, case-based problem solving is formalized as a search process, in which a solution space is traversed through the application of adaptation operators, and the choice of these operators is guided by case-based preferences.

In the following sections, there is a step by step description of the preference-based CBR framework.

## 3.2 Preference-based knowledge representation

In standard CBR, the experiences are stored in the form "solution $\boldsymbol{y}$ (optimally) solves problem $\boldsymbol{x}$". In preference-based CBR these experiences are replaced by information of the form "$\boldsymbol{y}$ is better (more preferred) than $\boldsymbol{z}$ as a solution for $\boldsymbol{x}$". Accordingly, the structure of the experiences we work with is symbolized in the form $\boldsymbol{y} \succeq_{\boldsymbol{x}} \boldsymbol{z}$ and suggests that, for the problem $\boldsymbol{x}$, the solution $\boldsymbol{y}$ is supposedly at least as good as $\boldsymbol{z}$.

To this end, it is by no means required that one of these solutions is optimal. It is worth mentioning, however, that knowledge about the optimality of a solution $\boldsymbol{y}^*$, if available, can be handled, too, as it simply means that $\boldsymbol{y}^* \succ \boldsymbol{y}$ for all $\boldsymbol{y} \neq \boldsymbol{y}^*$. In this sense, the conventional CBR setting can be considered as a special case of Pref-CBR.

The above idea of a preference-based approach to knowledge representation in CBR also suggests a natural extension of the case retrieval and inference steps, that is, the recommendation of solutions for a new query problem: Instead of just proposing a single solution, it would be desirable to predict a *ranking* of several (or even all) candidate solutions, ordered by their (estimated) degree of preference:

$$\boldsymbol{y}_1 \succeq_{\boldsymbol{x}} \boldsymbol{y}_2 \succeq_{\boldsymbol{x}} \boldsymbol{y}_3 \succeq_{\boldsymbol{x}} \cdots \succeq_{\boldsymbol{x}} \boldsymbol{y}_n \tag{3.1}$$

Obviously, the last problem mentioned above, namely the lack of guidance in the case of a failure, can thus be overcome.

As mentioned above, our Pref-CBR framework can handle working also in complex spaces. One such domain is the field of bioinformatics, in which structural databases storing information about geometrical and physicochemical properties of proteins are becoming increasingly important. Such databases can contain thousands of protein structures, where structural information is especially important for applications in computational chemistry and pharmacy, such as drug design. A functionality commonly offered by a structural database is similarity retrieval: Given a novel protein structure with unknown function, one is interested in finding similar proteins stored in the database. Somewhat simplified, a protein binding site or binding pocket can be thought of as a cavity on the surface of a protein in which important physicochemical reactions and interactions with other biomolecules are taking place, such as the binding of a small molecule (ligand) or the formation of a complex with another protein. Needless to say, binding sites are important targets for drug development. For that reason we will show this application as an example describing preference-based knowledge representation in Figure 3.1, where we can further understand the idea of preferences over solutions:

- *Which solution is better?* Showing two docking poses to a domain expert (chemist, pharmacist), she might be able to easily decide which of the molecules fits better. In this case, molecule B fits the protein more than molecule A, and that is why B is more preferred as a solution.

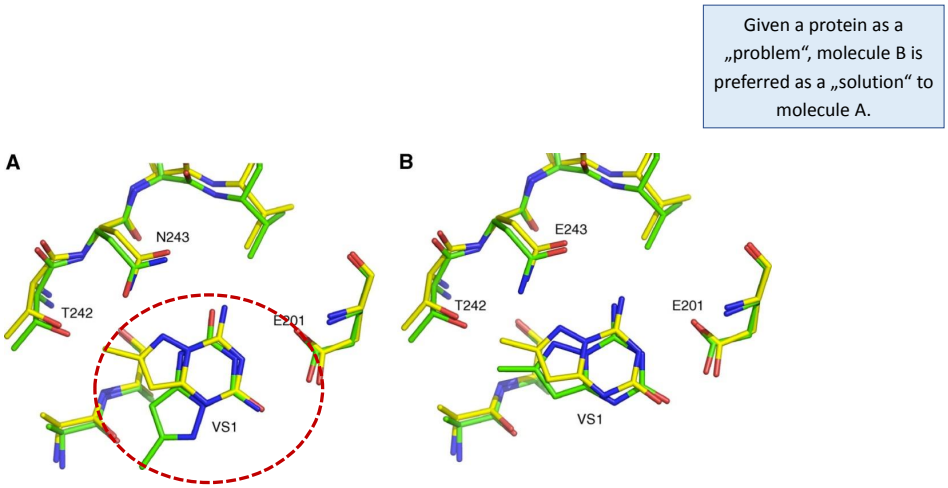


**Figure 3.1:** An Example of preference-based knowledge representation of experience

- *How good is each solution?* Chemist or pharmacist will find it difficult to assign a numerical score to an individual molecule.

- *What is the optimal solution?* The notion of *optimality* is not well defined, the space of molecules is huge and only partly known.

As we can see, this example shows how the use of preference-based CBR fits in this scenario where standard CBR cannot be used: an expert can give a preference of one solution over another, it is extremely difficult to give a quantitative answer and since the solution space is very complex, it is almost impossible to specify one *optimal* solution.

In order to realize an approach of that kind, a number of important questions need to be addressed, including the following: How to represent, organize and maintain case-based experiences, given in the form of preferences referring to a specific context, in an efficient way? How to select and access the experiences which are most relevant in a new problem solving situation? How to combine these experiences and exploit them to infer a solution or, more generally, a preference order on a set of candidate solutions, for the problem at hand? The answers to these questions will become clear as we progress with the explanation of the development of the Pref-CBR framework in the proceeding sections.

## 3.3 Formal setting and notation

In the following, we assume the problem space $\mathbb{X}$ to be equipped with a similarity measure $S_X : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_+$ or, equivalently, with a (reciprocal) distance measure $\Delta_X : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_+$. Thus, for any pair of problems $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{X}$, their similarity is denoted by $S_X(\boldsymbol{x}, \boldsymbol{x}')$ and their distance by $\Delta_X(\boldsymbol{x}, \boldsymbol{x}')$. Likewise, we assume the solution space $\mathbb{Y}$ to be equipped with a similarity measure $S_Y$ or, equivalently, with a (reciprocal) distance measure $\Delta_Y$. While the assumption of a similarity measure on problems is common in CBR, the existence of such a measure on the solution space is often not required. However, the latter is neither less natural than the former nor more difficult to define. In general, $\Delta_Y(\boldsymbol{y}, \boldsymbol{y}')$ can be thought of as a kind of adaptation cost, i.e., the (minimum) cost that needs to be invested to transform the solution $\boldsymbol{y}$ into $\boldsymbol{y}'$.

In Pref-CBR, problems $\boldsymbol{x} \in \mathbb{X}$ are not associated with single solutions but rather with preferences over solutions, that is, with elements from a class of preference structures $\mathfrak{P}(\mathbb{Y})$ over the solution space $\mathbb{Y}$. Here, we make the assumption that $\mathfrak{P}(\mathbb{Y})$ is given by the class of all weak order relations $\succeq$ on $\mathbb{Y}$, and we denote the relation associated with a problem $\boldsymbol{x}$ by $\succeq_{\boldsymbol{x}}$; recall that, from a weak order $\succeq$, a strict preference $\succ$ and an indifference $\sim$ are derived as follows: $\boldsymbol{y} \succ \boldsymbol{y}'$ iff $\boldsymbol{y} \succeq \boldsymbol{y}'$ and $\boldsymbol{y}' \not\succeq \boldsymbol{y}$, and $\boldsymbol{y} \sim \boldsymbol{y}'$ iff $\boldsymbol{y} \succeq \boldsymbol{y}'$ and $\boldsymbol{y}' \succeq \boldsymbol{y}$.

More precisely, we assume that $\succeq_{\boldsymbol{x}}$ has a specific form, which is defined by an "ideal" solution $\boldsymbol{y}^* \in \mathbb{Y}$ and the distance measure $\Delta_Y$: The closer a solution $\boldsymbol{y}$ to $\boldsymbol{y}^* = \boldsymbol{y}^*(\boldsymbol{x})$, the more it is preferred; thus, $\boldsymbol{y} \succeq_{\boldsymbol{x}} \boldsymbol{y}'$ iff $\Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*) \leq \Delta_Y(\boldsymbol{y}', \boldsymbol{y}^*)$. Please note that, when starting from an order relation $\succeq_{\boldsymbol{x}}$, then the existence of an "ideal" solution is in principle no additional assumption (since a weak order has a maximal element, at least if the underlying space is topologically closed). Instead, the additional assumption we make is that the order relations $\succeq_{\boldsymbol{x}}$ and $\succeq_{\boldsymbol{x}'}$ associated with different problems $\boldsymbol{x}$ and $\boldsymbol{x}'$ have a common structure, which is determined by the distance measure $\Delta_Y$. In conjunction with the regularity assumption that is commonly made in CBR, namely that similar problems tend to have similar (ideal) solutions, this property legitimates a preference-based version of this assumption: *Similar problems are likely to induce similar preferences over solutions.*

Figure 3.2 illustrates the idea of preference-based CBR problem solving procedure and how a set of preferences on solutions are generated by the oracle. To solve a

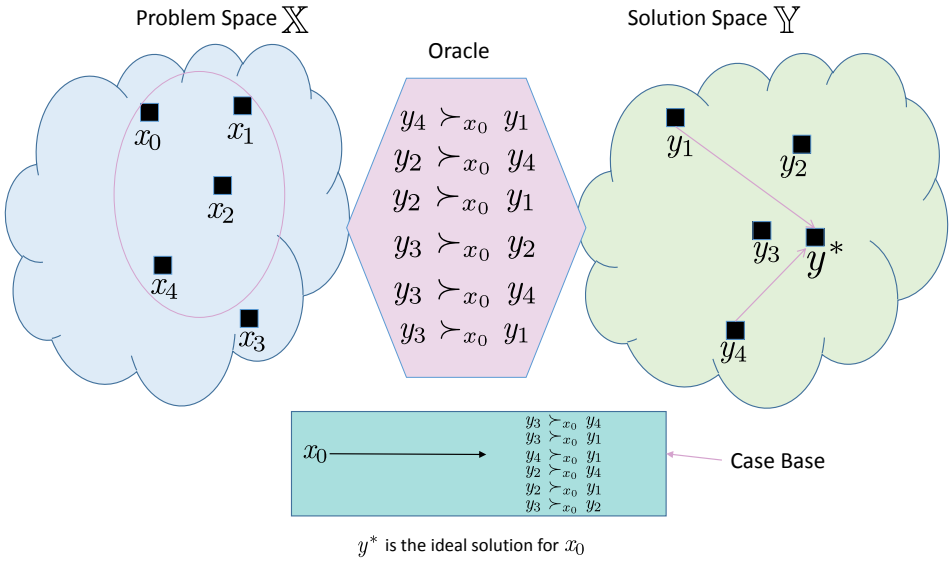


**Figure 3.2:** An example of problem-solving in Pref-CBR

case $\boldsymbol{x}_0$, similar problems are retrieved, in this example these are $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{x}_4$. The preferences of those nearest neighbors are used to find an initial solution for $\boldsymbol{x}_0$, the details of this process will be explained in the proceeding sections. Having this computed initial solution is the starting point for our oracle to come into action. The oracle starts to generate preferences based on the idea of a solution being preferred over another solution depending on how close it is to the *ideal* solution, in this example being $\boldsymbol{y}^*$. In our example, $\boldsymbol{y}_4 \succeq_{\boldsymbol{x}_0} \boldsymbol{y}_1$ because $\Delta_Y(\boldsymbol{y}_4, \boldsymbol{y}^*) \leq \Delta_Y(\boldsymbol{y}_1, \boldsymbol{y}^*)$. Up to this point, the idea now is to understand the concept of preferences on solutions and the basis on which the preferences are created by our oracle. After each query to the oracle, a new candidate solution is provided and at the end of the cycle of queries to the oracle a final solution is obtained and the search process ends. The problem $\boldsymbol{x}_0$ is stored in the case base along with its solution which consists of a set of preferences generated by the oracle, as shown in Figure 3.2.

As seen from the example of problem-solving in Pref-CBR, after a case is solved it is stored in the case base along with its solution which is a set of preferences over solutions as well as a final solution. In standard CBR, after a case is solved it is stored in the case base along with its final solution only which is a single solution. It is important to consider that this is a major distinction between our Pref-CBR problem-solving

framework and standard CBR problem-solving.

## 3.4 CBR as preference-guided search

Case-based reasoning and (heuristic) search can be connected in various ways. One idea is to exploit CBR in order to enhance heuristic search, which essentially comes down to using case-based experience to guide the search behavior [64, 65, 66]. The other way around, the CBR process itself can be formalized as a search process, namely a traversal of the space of potential solutions [31]. This idea is quite appealing: On the one side, it is close to practical, human-like problem solving, which is indeed often realized as a kind of trial-and-error process, in which a candidate solution is successively modified and improved until a satisfactory solution is found. On the other side, this idea is also amenable to a proper formalization and automation, since *searching* is what computers are really good at; besides, heuristic search is one of the best developed subfields of AI.

Needless to say, both directions (enhancing search through CBR and formalizing CBR as search) are not mutually exclusive and can be combined with each other. In our approach, this is accomplished by implementing case-based problem solving as a search process that is guided by preference information collected in previous problem solving episodes. The type of application we have in mind is characterized by two important properties:

- *The evaluation of candidate solutions is expensive.* Therefore, only relatively few candidates can be considered in a problem solving episode before a selection is made. Typical examples include cases where an evaluation requires time-consuming simulation studies or human intervention. In the cooking domain, for example, the evaluation of a recipe may require its preparation and tasting. Needless to say, this can only be done for a limited number of variations.

- *The quality of candidate solutions is difficult to quantify.* Therefore, instead of asking for numerical utility degrees, we make a much weaker assumption: Feedback is only provided in the form of pairwise comparisons, informing about which of two candidate solutions is preferred (for example, which of two meals tastes better). Formally, we assume the existence of an oracle (for example, a user or a computer program) as explained above in the introduction of the chapter. The

aforementioned oracle, when given a problem $\boldsymbol{x}_0$ and two solutions $\boldsymbol{y}$ and $\boldsymbol{z}$ as input, returns a preference $\boldsymbol{y} \succ \boldsymbol{z}$ or $\boldsymbol{z} \succ \boldsymbol{y}$ (or perhaps also an indifference $\boldsymbol{y} \sim \boldsymbol{z}$) as output.

We assume the solution space $\mathbb{Y}$ to be equipped with a topology that is defined through a *neighborhood structure*: For each $\boldsymbol{y} \in \mathbb{Y}$, we denote by $\mathcal{N}(\boldsymbol{y}) \subseteq \mathbb{Y}$ the neighborhood of this candidate solution. The neighborhood is thought of as those solutions that can be produced through a single modification of $\boldsymbol{y}$, i.e., by applying one of the available adaptation operators to $\boldsymbol{y}$ (for example, adding or removing a single ingredient in a recipe). Since these operators are application-dependent, we are not going to specify them further here.

Our case base **CB** stores problems $\boldsymbol{x}_i$ together with a set of preferences $\mathcal{P}(\boldsymbol{x}_i)$ that have been observed for these problems. Thus, each $\mathcal{P}(\boldsymbol{x}_i)$ is a set of preferences of the form $\boldsymbol{y} \succ_{\boldsymbol{x}_i} \boldsymbol{z}$. As will be explained further below, these preferences are collected while searching for a good solution to $\boldsymbol{x}_i$.

We conceive preference-based CBR as an iterative process in which problems are solved one by one; our current implementation of this process is described in pseudo-code in Algorithm 1. In each problem solving episode, a good solution for a new query problem is sought, and new experiences in the form of preferences are collected. In what follows, we give a high-level description of a single problem solving episode (lines 5–23 of the algorithm):

- Given a new query problem $\boldsymbol{x}_0$, the $K$ nearest neighbors[1] $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K$ of this problem (i.e., those with smallest distance in the sense of $\Delta_X$) are retrieved from the case base **CB**, together with their preference information $\mathcal{P}(\boldsymbol{x}_1), \ldots, \mathcal{P}(\boldsymbol{x}_K)$.

- This information is collected in a single set of preferences $\mathcal{P}$, which is considered representative for the problem $\boldsymbol{x}_0$ and used to guide the search process (line 8).

- The search for a solution starts with a initial candidate $\boldsymbol{y}^* \in \mathbb{Y}$ chosen at random (line 9) and iterates $L$ times. Restricting the number of iterations of the queries to the oracle by an upper bound $L$ reflects our assumption that an evaluation of a candidate solution is costly.

[1] As long as the case base contains less than $K$ cases, all these cases are taken.

- In each iteration, a new candidate $\boldsymbol{y}^{query}$ is determined and given as a query to the oracle (line 15), i.e., the oracle is asked to compare $\boldsymbol{y}^{query}$ with the current best solution $\boldsymbol{y}^*$ (line 16). The preference reported by the oracle is memorized by adding it to the preference set $\mathcal{P}_0 = \mathcal{P}(\boldsymbol{x}_0)$ associated with $\boldsymbol{x}_0$ (line 17), as well as to the set $\mathcal{P}$ of preferences used for guiding the search process. Moreover, the better solution is retained as the current best candidate (line 18).

- When the search stops, the current best solution $\boldsymbol{y}^*$ is returned, and the case $(\boldsymbol{x}_0, \mathcal{P}_0)$ is added to the case base.

The preference-based guidance of the search process is realized in lines 9 and 14–15. Here, the case-based inference method, see 3.3, (referred to as CBI in the pseudo-code) described in Section 3.5, is used to find the most promising candidate among the neighborhood of the current solution $\boldsymbol{y}^*$ (excluding those solutions that have already been tried). By providing information about which of these candidates will most likely constitute a good solution for $\boldsymbol{x}_0$, it (hopefully) points the search into the most promising direction. Please note that in line 15, case-based inference is not applied to the whole set of preferences $\mathcal{P}$ collected so far, but only to a subset of the $J$ preferences $\mathcal{P}^{nn}$ that are closest (and hence most relevant) to the current search state $\boldsymbol{y}^*$; here, the distance between a preference $\boldsymbol{y} \succ \boldsymbol{z}$ and a solution $\boldsymbol{y}^*$ is defined as

$$\Delta\left(\boldsymbol{y}^*, \boldsymbol{y} \succ \boldsymbol{z}\right) = \min\left\{\Delta_Y\left(\boldsymbol{y}^*, \boldsymbol{y}\right), \Delta_Y\left(\boldsymbol{y}^*, \boldsymbol{z}\right)\right\} \quad , \tag{3.2}$$

i.e., the preference is considered relevant if either $\boldsymbol{y}$ is close to $\boldsymbol{y}^*$ or $\boldsymbol{z}$ is close to $\boldsymbol{y}^*$. This is done in order to allow for controlling the locality of the search: The smaller $J$, the less preferences are used, i.e., the more local the determination of the direction of the search process[1] becomes (by definition, CBI returns a random element from $\mathbb{Y}^{nn}$ if $\mathcal{P}^{nn} = \emptyset$, i.e., if $J = 0$). Note that, if $J = 1$, then only the preference that has been added in the last step is looked at (since this preference involves $\boldsymbol{y}^*$, and therefore its distance according to (3.2) is 0). Thus, search will move ahead in the same direction if the last modification has led to an improvement, and otherwise reverse its direction. In general, a larger $J$ increases the bias of the search process and makes it more "inert". This is advantageous if the preferences coming from the neighbors of $\boldsymbol{x}_0$ are indeed

[1]The term "direction" is used figuratively here; if $\mathbb{Y}$ is not a metric space, there is not necessarily a direction in a strictly mathematical sense.

**Algorithm 1** Pref-CBR Search($K$, $L$, $J$)

**Require:** $K$ = number of nearest neighbors collected in the case base
$L$ = total number of queries to the oracle
$J$ = number of preferences used to guide the search process

$\mathbb{X}_0 \leftarrow$ list of problems to be solved ▷ a subset of $\boldsymbol{X}$
$Q \leftarrow [\cdot]$ ▷ empty list of performance degrees
**CB** $\leftarrow \emptyset$ ▷ initialize empty case base
**while** $\mathbb{X}_0$ not empty **do**
  $\boldsymbol{x}_0 \leftarrow$ pop first element from $\mathbb{X}_0$ ▷ new problem to be solved
  $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\} \leftarrow$ nearest neighbors of $\boldsymbol{x}_0$ in **CB** (according to $\Delta_X$)
  $\{\mathcal{P}(\boldsymbol{x}_1), \ldots, \mathcal{P}(\boldsymbol{x}_K)\} \leftarrow$ preferences associated with nearest neighbors
  $\mathcal{P} \leftarrow \mathcal{P}(\boldsymbol{x}_1) \cup \mathcal{P}(\boldsymbol{x}_2) \cup \ldots \cup \mathcal{P}(\boldsymbol{x}_k)$ ▷ combine neighbor preferences
  $\boldsymbol{y}^* \leftarrow \mathrm{CBI}(\mathcal{P}, \mathbb{Y})$ ▷ select an initial candidate solution
  $\mathbb{Y}^{vis} \leftarrow \{\boldsymbol{y}^*\}$ ▷ candidates already visited
  $\mathcal{P}_0 \leftarrow \emptyset$ ▷ initialize new preferences
  **for** $i = 1$ to $L$ **do**
    $\mathcal{P}^{nn} = \{\boldsymbol{y}^{(j)} \succ \boldsymbol{z}^{(j)}\}_{j=1}^{J} \leftarrow J$ preferences in $\mathcal{P} \cup \mathcal{P}_0$ closest to $\boldsymbol{y}^*$
    $\mathbb{Y}^{nn} \leftarrow$ neighborhood $\mathcal{N}(\boldsymbol{y}^*)$ of $\boldsymbol{y}^*$ in $\mathbb{Y} \setminus \mathbb{Y}^{vis}$
    $\boldsymbol{y}^{query} \leftarrow \mathrm{CBI}(\mathcal{P}^{nn}, \mathbb{Y}^{nn})$ ▷ find next candidate
    $[\boldsymbol{y} \succ \boldsymbol{z}] \leftarrow \mathrm{Oracle}(\boldsymbol{x}_0, \boldsymbol{y}^{query}, \boldsymbol{y}^*)$ ▷ check if new candidate is better
    $\mathcal{P}_0 \leftarrow \mathcal{P}_0 \cup \{\boldsymbol{y} \succ \boldsymbol{z}\}$ ▷ memorize preference
    $\boldsymbol{y}^* \leftarrow \boldsymbol{y}$ ▷ adopt the current best solution
    $\mathbb{Y}^{vis} \leftarrow \mathbb{Y}^{vis} \cup \{\boldsymbol{y}^{query}\}$
  **end for**
  $q \leftarrow$ performance of solution $\boldsymbol{y}^*$ for problem $\boldsymbol{x}_0$
  $Q \leftarrow [Q, q]$ ▷ store the performance
  **CB** $\leftarrow$ **CB** $\cup \{(\boldsymbol{x}_0, \mathcal{P}_0)\}$ ▷ memorize new experience
**end while**
return list $Q$ of performance degrees

representative and, therefore, are pointing in the right direction. Otherwise, of course, too much reliance on these preferences may prevent one from searching in other regions of the solution space that might be more appropriate for $\boldsymbol{x}_0$.

Let us mention that a stochastic component can be added to our search procedure in a quite natural way, this can be done at a later stage, as an extension to the work finished in this thesis. To this end, the case-based inference procedure CBI simply returns one of the candidate solutions $\boldsymbol{y} \in \mathbb{Y}^{cand}$ with a probability that is proportional to the corresponding likelihood degrees of these solutions (instead of deterministically choosing the solution with the highest likelihood). In the proceeding section, a detailed description of the functionality of the case-based inference method is provided.

## 3.5 Case-based inference

The key idea of Pref-CBR is to exploit experience in the form of previously observed preferences, deemed relevant for the problem at hand, in order to support the current problem solving episode; like in standard CBR, the *relevance* of a preference will typically be decided on the basis of problem similarity, i.e., those preferences will be deemed relevant that pertain to similar problems. An important question that needs to be answered in this connection is the following: Given a set of observed preferences on solutions, considered representative for a problem $\boldsymbol{x}_0$, what is the underlying preference structure $\succeq_{\boldsymbol{x}_0}$ or, equivalently, what is the most likely "ideal" solution $\boldsymbol{y}^*$ for $\boldsymbol{x}_0$?

### 3.5.1 Case-based inference as probability estimation

We approach this problem from a statistical perspective, considering the true preference model $\succeq_{\boldsymbol{x}_0} \in \mathfrak{P}(\mathbb{Y})$ associated with the query $\boldsymbol{x}_0$ as a random variable $Z$ with distribution $\mathbf{P}(\cdot \mid \boldsymbol{x}_0)$, where $\mathbf{P}(\cdot \mid \boldsymbol{x}_0)$ is a distribution $\mathbf{P}_\theta(\cdot)$ parametrized by $\theta = \theta(\boldsymbol{x}_0) \in \Theta$. The problem is then to estimate this distribution or, equivalently, the parameter $\theta$ on the basis of the information available. This information consists of a set $\mathcal{D}$ of preferences of the form $\boldsymbol{y} \succ \boldsymbol{z}$ between solutions.

The basic assumption underlying nearest neighbor estimation is that the conditional probability distribution of the output given the input is (approximately) locally constant, that is, $\mathbf{P}(\cdot \mid \boldsymbol{x}_0) \approx \mathbf{P}(\cdot \mid \boldsymbol{x})$ for $\boldsymbol{x}$ close to $\boldsymbol{x}_0$. Thus, if the above preferences are coming from problems $\boldsymbol{x}$ similar to $\boldsymbol{x}_0$ (namely from the nearest neighbors of $\boldsymbol{x}_0$ in

the case base), then this assumption justifies considering $\mathcal{D}$ as a representative sample of $\mathbf{P}_\theta(\cdot)$ and, hence, estimating $\theta$ via maximum likelihood (ML) by

$$\theta^{ML} = \arg\max_{\theta\in\Theta} \mathbf{P}_\theta(\mathcal{D}) \ . \tag{3.3}$$

An important prerequisite for putting this approach into practice is a suitable data generating process, i.e., a process generating preferences in a stochastic way.

### 3.5.2 A discrete choice model

Our data generating process is based on the idea of a discrete choice model as used in choice and decision theory [67]. Recall that the (absolute) preference for a solution $\boldsymbol{y} \in \mathbb{Y}$ supposedly depends on its distance $\Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*) \geq 0$ to an "ideal" solution $\boldsymbol{y}^*$, where $\Delta(\boldsymbol{y}, \boldsymbol{y}^*)$ can be seen as a "degree of suboptimality" of $\boldsymbol{y}$. As explained in [12], more specific assumptions on an underlying (latent) utility function on solutions justify the *logit* model of discrete choice:

$$\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z}) = \Big(1 + \exp\big(-\beta(\Delta_Y(\boldsymbol{z}, \boldsymbol{y}^*) - \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*))\big)\Big)^{-1} \tag{3.4}$$

Thus, the probability of observing the (revealed) preference $\boldsymbol{y} \succ \boldsymbol{z}$ depends on the degree of suboptimality of $\boldsymbol{y}$ and $\boldsymbol{z}$, namely their respective distances to the ideal solution, $\Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)$ and $\Delta_Y(\boldsymbol{z}, \boldsymbol{y}^*)$: The larger the difference $\Delta_Y(\boldsymbol{z}, \boldsymbol{y}^*) - \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)$, i.e., the less optimal $\boldsymbol{z}$ in comparison to $\boldsymbol{y}$, the larger the probability to observe $\boldsymbol{y} \succ \boldsymbol{z}$; if $\Delta_Y(\boldsymbol{z}, \boldsymbol{y}^*) = \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)$, then $\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z}) = 1/2$. The coefficient $\beta$ can be seen as a measure of reliability of the preference feedback. For large $\beta$, the probability (3.4) converges toward 0 if $\Delta_Y(\boldsymbol{z}, \boldsymbol{y}^*) < \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)$ and toward 1 if $\Delta_Y(\boldsymbol{z}, \boldsymbol{y}^*) > \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)$; this corresponds to a deterministic (error-free) information source. The other extreme case, namely $\beta = 0$, models a completely unreliable source reporting preferences at random.

The graphical illustration of the probabilistic model in Figure 3.3, shows how the probability (3.4) to observe $\boldsymbol{y} \succ \boldsymbol{y}'$ converges toward 0 if $\Delta_Y(\boldsymbol{y}', \boldsymbol{y}^*) < \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)$ and toward 1 if $\Delta_Y(\boldsymbol{y}', \boldsymbol{y}^*) > \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)$, given precise information (high value of $\beta$). On the contrary, given less reliable observations of the preferences ($\beta = 0$), preferences are reported at random and the value of the probability approximately equals 0.5.

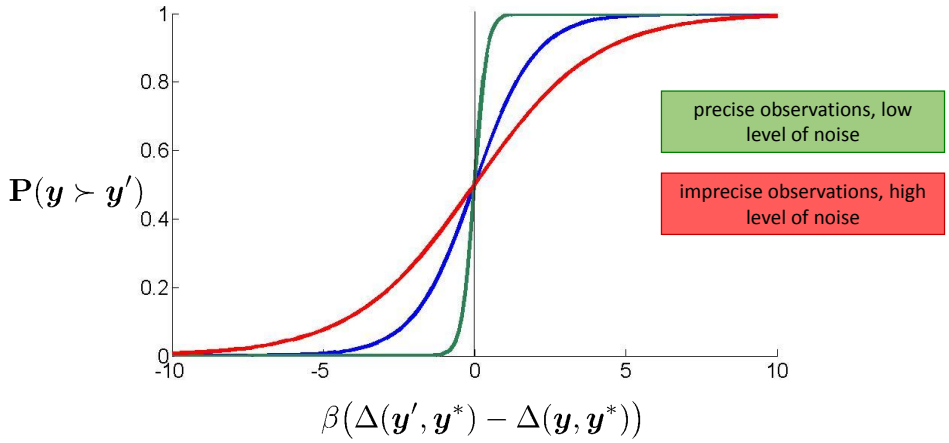


**Figure 3.3:** A Discrete Choice Model for preferences on solutions

### 3.5.3 Maximum likelihood estimation

The probabilistic model outlined above is specified by two parameters: the ideal solution $\boldsymbol{y}^*$ and the (true) precision parameter $\beta^* \in \mathbb{R}_+$. Depending on the context in which these parameters are sought, the ideal solution might be unrestricted (i.e., any element of $\mathbb{Y}$ is an eligible candidate), or it might be restricted to a certain subset $\mathbb{Y}_0 \subseteq \mathbb{Y}$ of candidates.

Now, to estimate the parameter vector $\theta^* = (\boldsymbol{y}^*, \beta^*) \in \mathbb{Y}_0 \times \mathbb{R}^*$ from a given set $\mathcal{D} = \{\boldsymbol{y}^{(i)} \succ \boldsymbol{z}^{(i)}\}_{i=1}^N$ of observed preferences, we refer to the maximum likelihood (ML) estimation principle. Assuming independence of the preferences, the log-likelihood of $\theta = (\boldsymbol{y}, \beta)$ is given by

$$\ell(\theta) = \ell(\boldsymbol{y}, \beta) = -\sum_{i=1}^{N} \log\Big(1 + \exp\big(-\beta(\Delta(\boldsymbol{z}^{(i)}, \boldsymbol{y}) - \Delta(\boldsymbol{y}^{(i)}, \boldsymbol{y}))\big)\Big). \tag{3.5}$$

The maximum likelihood estimation (MLE) $\theta_{ML} = (\boldsymbol{y}^{ML}, \beta^{ML})$ of $\theta^*$ is given by the maximizer of (3.5):

$$\theta_{ML} = \big(\boldsymbol{y}^{ML}, \beta^{ML}\big) = \arg \max_{\boldsymbol{y} \in \mathbb{Y}_0, \beta \in \mathbb{R}_+} \ell(\boldsymbol{y}, \beta)$$

It is important to note that in our search procedure, we form a neighborhood around an initial solution $\boldsymbol{y}$, this neighborhood can be in whichever form we choose depending on the structure of the solution space. If we have a continuous solution space, we can

form for example a circular neighborhood or form a gaussian distribution neighborhood around the initial solution. If the solution space is discrete then we form a neighborhood around the initial solution which consists of candidates differing by one discrete step from the initial solution (for an item set it would be item sets which have one item added or one item removed, for a permutation it would be switching orders of the permutation, etc.). Using (3.5) and our set of observed preferences $\mathcal{D} = \{\boldsymbol{y}^{(i)} \succ \boldsymbol{z}^{(i)}\}_{i=1}^{N}$, we compute the likelihoods of the neighborhood candidates in the subset $\mathbb{Y}_0 \subseteq \mathbb{Y}$ and thus find the solution $\boldsymbol{y}^*$ with the maximum likelihood. The maximum likelihood solution is then our candidate solution and we form a suitable neighborhood around it and start to ask the oracle again. At the end of the cycle of querying the oracle, the last obtained preferred solution is considered our best obtained solution for the problem at hand. This solution is then stored in the case base as our optimal solution for the problem. The preferences generated during the cycle of querying the oracle during the problem-solving episode are also stored in the case base along with the problem for later reuse when a new problem is solved. For simplicity we fix $\beta$ and therefore we can then easily determine our $\boldsymbol{y}^*$.

## 3.6 Conclusion

In this chapter, we have presented a general framework for CBR in which experience is represented in the form of contextualized preferences, and these preferences are used to direct an adaptive problem solving process that is formalized as a search procedure. This kind of preference-based CBR is an interesting alternative to conventional CBR whenever solution quality is a matter of degree and feedback is only provided in an indirect or qualitative way. The effectiveness of our generic framework is illustrated in several concrete case studies, presented in Chapter 7.

The Pref-CBR framework will be generalized and extended in two directions in the next two chapters. First, as already mentioned, the similarity (distance) measure in the solution space has an important influence on the preference relations $\succeq_{\boldsymbol{x}}$ associated with problems $\boldsymbol{x} \in \mathbb{X}$ and essentially determines the structure of these relations (cf. Section 3.3). Therefore, we show that a proper specification of this measure enhances the effectiveness of our preference-guided search procedure. Accordingly, it would be desirable to allow for a data-driven adaptation of this measure, that is, to enable

the CBR system to adapt this measure whenever it does not seem to be optimal. We propose a method for learning similarity measures in the solution space from qualitative feedback, which appears to be ideally suited for this purpose. For further optimizing the performance of our Pref-CBR framework, we also propose another method for learning similarity measures in the problem space by learning from examples. The aforementioned methods are described in detail in the next chapter.

As the number of preferences collected over the course of time may become rather large, we also develop effective methods for case base maintenance, which specifically suit the Pref-CBR framework. We propose some case base maintenance strategies which allow us to increase the efficiency of the case base while maintaining its performance. We propose two directions of case base maintenance strategies, inter-case maintenance and intra-case maintenance. The former maintenance methods handle whole cases, while the latter methods handle preferences (parts of cases). In Chapter 5, we will describe some case base maintenance strategies which specifically suit our framework and enhance its efficiency.

After describing in detail how our Pref-CBR framework operates, we will learn how the integrated components of learning similarity measures as well as the case base maintenance strategies are embedded in the framework. We will also show how they affect the performance and efficiency of the whole system. We will also look at other methods and see how they are similar or different from our framework. In Chapter 6, we discuss some methodologies which are related to our Pref-CBR framework. These methodologies include different search methods and some machine learning approaches. We compare the different approaches with our Pref-CBR framework, discuss the similarities and the differences and convey the position in which our approach is situated amongst the other approaches which are related to ours.

# 4

# Learning Similarity Measures in Pref-CBR

In our Pref-CBR framework, case-based problem solving is formalized as a preference-guided search process in the space of candidate solutions, which is equipped with a similarity (or, equivalently, a distance) measure. A well-defined similarity measure is crucial for the optimal performance of a case-based reasoning system. In preference-based CBR, the preferences induced during the search procedure can be used to learn the similarity measures and thus lead to improved search performance.

Like many other CBR approaches, Pref-CBR proceeds from a formal framework consisting of a problem space $\mathbb{X}$ and a solution space $\mathbb{Y}$. Yet, somewhat less common, it assumes a similarity (or distance) measure to be defined not only on $\mathbb{X}$ but also on $\mathbb{Y}$. Moreover, it assumes a strong connection between the notions of *preference* and *similarity*. More specifically, for each problem $\boldsymbol{x} \in \mathbb{X}$, it assumes the existence of a theoretically *ideal* solution $\boldsymbol{y}^* \in \mathbb{Y}$ (even if this solution might be fictitious and cannot be materialized), and the less another solution $\boldsymbol{y}$ differs from $\boldsymbol{y}^*$ in the sense of a distance measure $\Delta_Y$, the more this solution is preferred.

As a consequence, the performance and effectiveness of Pref-CBR is strongly influenced by the distance measure $\Delta_Y$: The better this measure captures the true differences between solutions, the more effective Pref-CBR will be. In this chapter, we therefore extend our framework through the integration of a *distance learning* module. Thus, the idea is to make use of the experience collected in a problem solving episode, not only to extend the case base through memorization of preferences, but also to adapt

the distance measure $\Delta_Y$. Since the efficacy of Pref-CBR is influenced by the adequacy of this measure, we propose a learning method for adapting solution similarity on the basis of experience gathered by the CBR system over the course of time. More specifically, our solution similarity learning method makes use of an underlying probabilistic model and realizes adaptation as Bayesian inference.

The importance of distance metric learning in optimizing the performance of many learning and data mining algorithms has been mentioned in the work of [68]. An important challenge we had to consider in formalizing our learning method, is the incorporation of prior information of paired comparisons [69] through our search procedure. Existing approaches for learning distance metrics from pairwise comparisons suffer from either being unreliable when the number of training examples is small [68], or these methods often use ad-hoc algorithms with little or no formal basis [69]. Due to these factors, the Bayesian approach is our method for learning similarity measures in the solution space. Our aim is to implement a formal and accurate learning method, which actively uses prior as well as current preference information to yield a posterior distribution that increases the probability of choosing an optimal solution for a current problem at hand.

To further optimize the performance of our Pref-CBR framework, we pursue learning similarity measures also in the problem space. We use the well-known perceptron algorithm to combine given local similarity measures and learn how to combine them into a global measure. This method elicits global similarity measures on the basis of feedback in the form of positive and negative examples to be used for learning. We learn the similarity measures from qualitative feedback: given a reference case and two cases to compare with, we see which of these two cases is more similar to the reference case. The general idea of this approach basically reduces the problem of distance learning to a binary classification problem.

## 4.1 Learning similarity measures in the solution space

The learning and adaptation of similarity or distance measures has been studied intensively in the literature, not only in CBR but also in related fields like machine learning. Yet, our approach has a number of properties that distinguish it from most others: similarity is learned in the solution space, not only in the problem space; training

information is purely qualitative and based on paired comparisons. Learning is done within the framework of Bayesian inference, making use of a probabilistic model. In this section, we will explain in further detail our proposed method for learning similarity measures in the solution space.

### 4.1.1 Related work

In this subsection, we will mention previous work on emphasizing the importance of learning similarity metrics on the performance of a system, introducing the Bayesian approach for learning in different applications. Some work is also mentioned about using preferences for learning of similarity measures as well as how interest was initiated for learning similarity measures in the solution space and not exclusively in the problem space. In our framework, we use our generated preferences in the Bayesian approach for learning the similarity metrics in the solution space and we also show the efficacy of learning on the search performance. Thus we find it important to discuss some previous work on all these different aspects.

Considering the importance of the concept of similarity in CBR, much work has been done in learning how to improve a CBR system's performance by focusing more deeply on this similarity concept and learning its metrics. In practice, similarity measures are used to compute the similarity between queries and cases and thus their basic task is namely the retrieval of useful cases [70]. As the evaluated similarity values reflect the utility or the appropriateness of solutions of the known cases, they offer important information to be utilized in the next step of the choice for a solution for the problem in query [71]. In other words, since CBR systems retrieve cases using a similarity function, there is a degradation in the precision of this similarity function when there are irrelevant features or if the data is noisy and unreliable [72]. It is then useful to identify as much of the irrelevant information as possible by local feature selection or weighting methods. It follows then to say that if a correct classification occurs then the weights of the matching features are incremented, while those of mismatching features are decremented of the new query, by a fixed amount [73]; in our framework we can say if a correct choice of a solution is made (one that complies with the choice of the oracle) then the weights of the matching local features are incremented while those of mismatching local features are decremented.

Appropriate metrics are also essential for the performance of distance-based methods such as nearest neighbor estimation, which are used for classification, regression, and related problems. Metric learning has therefore been studied quite intensively in machine learning and pattern recognition. While Mahalanobis distance metric learning has received specific attention in this regard, more involved problems such as nonlinear metric learning, local metric learning, semi-supervised metric learning, and metric learning for structured data have been tackled more recently. We refer to [74] for a comprehensive and up-to-date survey of the metric learning literature.

The idea of the formulation of a Bayesian approach for learning from paired comparisons has been previously introduced in [75]. The Bayesian method has also been used for example in optimizing pairing methods, to be used in the design of tournament schedules for players of games and sports [69]. Similar to the work discussed on preference elicitation, the authors in [76] propose using a Bayesian model for the querying process to learn control policies through trajectory preference queries to an expert. A common criterion between this previously mentioned work and ours, is the goal of finding an optimal target policy (in our case an optimal solution) from the expert with as few queries as possible. A Bayesian approach to distance metric learning has also been proposed in [77]. Here, the authors estimate a posterior distribution for the distance metric from labeled pairwise constraints, namely equivalence constraints (pairs of similar objects) and inequivalence constraints (pairs of dissimilar objects). Worth mentioning is also the Bayesian approach to preference elicitation by [78]. Although it is concerned with utility instead of distance learning, the authors proceed from training information in the form of paired comparisons, and assume preferences to be generated by the Bradley-Terry model. Their model is still a bit simpler than our model (4.2) and permits the derivation of closed-form Bayesian updates (using a suitable family of conjugate priors).

Using preferences for learning similarity metrics has been discussed in [79]; the dissimilarities between preferences are measured using Kemeny distance, allowing for quantifying disagreements according to where they occur in preferences. It is worthy to note that the preferences which are measured in the latter work are in the form of triplets, as well as being strict. Learning similarity measures on preference structures has also been pursued in various setups. In [80], the authors try to predict a user's preferences from other users' preferences by defining some distance measure on preference

orders. In their work, after getting some preferences from the new user, the defined distance measures are applied to retrieve the closest matching preference structure to the current user regardless of the order of questions.

Similar to the idea of rescaling or re-weighting the input space relative to the observed user's preferences on plans in the work of [81], in our work we aim to rescale or re-weight the output space relative to the observed preferences of the users as we progress with our search process. This should lead to an improved search performance as the case base grows and accordingly lead to reaching an optimal solution faster. In our Pref-CBR framework, we use contextual partial pairwise preferences to learn our similarity measures from qualitative feedback. Continuing on the work of [17], we got the motivation to use a method that learns how to combine given local similarity measures into a global one within our Pref-CBR framework; given qualitative feedback in the form of similarity comparisons, we try to learn the underlying similarity (distance) measure by developing a machine learning approach for similarity assessment in the solution space.

Similarity learning in CBR has almost exclusively focused on learning similarity in the problem space. This is also true for the work of Stahl [70, 82, 83, 84], which nevertheless share a number of commonalities with our approach. In particular, he also considers the learning of weights in a linear combination of local similarity functions [82, 85], albeit based on different types of training information and using other learning techniques. Although defining adequate similarity measures is one of the most crucial tasks when developing CBR applications, it is a difficult task and, unfortunately, it has been supported by a limited number of machine learning techniques [82]. In addition to choosing and maintaining an appropriate set of feature weights in a case base, it should also be considered that accordingly, the relative importance of the cases is changing with time [86]. Therefore, we also have the aim to use a learning method that is able to cope with the continuous evolution of a case base and accordingly, adjust the feature weights as the case base grows.

As we have mentioned before, our framework has the advantage over other methods of working with more complex structures of problem/solution spaces. As an example, let us assume we have our solutions composed of features, but of different concepts such as time, cost and location. The time could consist of two features (initial and final), our cost of two features (running cost and profit) and our location consisting of

three features (coordinates). We can say that we have in general for this example three independent concepts, which we can choose to be our similarity measures between the different solutions. Let us now assume that we defined three features for measuring similarity (or distance) between solutions, each of these three features would be the absolute value of differences of one group. Using the Bayesian learning method now allows for learning from the generated preferences how to balance between these three independent concepts, as well as recognizing which concept is more important and adjusting the weights of each of these concepts according to its significance deduced from the oracle. After reading the rest of this section, it will become more clear how the Bayesian learning method is implemented and how it supports our model which can handle simple as well as complex spaces.

### 4.1.2 Pref-CBR formal framework for learning similarity measures

Recalling from the previous chapter, we assume the problem space $\mathbb{X}$ to be equipped with a similarity measure $S_X : \mathbb{X}\times\mathbb{X} \to \mathbb{R}_+$ or, equivalently, with a (reciprocal) distance measure $\Delta_X : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_+$. Thus, for any pair of problems $\boldsymbol{x}, \boldsymbol{x'} \in \mathbb{X}$, their similarity is denoted by $S_X(\boldsymbol{x}, \boldsymbol{x'})$ and their distance by $\Delta_X(\boldsymbol{x}, \boldsymbol{x'})$. Likewise, we assume the solution space $\mathbb{Y}$ to be equipped with a similarity measure $S_Y$ or, equivalently, with a (reciprocal) distance measure $\Delta_Y$.

In general, $\Delta_Y(\boldsymbol{y}, \boldsymbol{y'})$ can be thought of as a kind of adaptation cost, i.e., the (minimum) cost that needs to be invested to transform the solution $\boldsymbol{y}$ into $\boldsymbol{y'}$. As will become clear later on, our framework suggests a natural connection between distance and similarity, which involves a parameter $\beta \geq 0$ and is of the following form:

$$S_Y(\boldsymbol{y}, \boldsymbol{y'}) = \exp\big(-\beta \cdot \Delta_Y(\boldsymbol{y}, \boldsymbol{y'})\big) \in (0, 1] \tag{4.1}$$

Recall also that the (absolute) preference for a solution $\boldsymbol{y} \in \mathbb{Y}$ supposedly depends on its distance $\Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*) \geq 0$ to an ideal solution $\boldsymbol{y}^*$, where $\Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)$ can be seen as a "degree of suboptimality" of $\boldsymbol{y}$.

More specific assumptions on an underlying (latent) utility function on solutions justify the *logit* model of discrete choice, explained in detail in Chapter 3:

$$\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z}) = \mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^*) = \frac{S_Y(\boldsymbol{y}, \boldsymbol{y}^*)}{S_Y(\boldsymbol{y}, \boldsymbol{y}^*) + S_Y(\boldsymbol{z}, \boldsymbol{y}^*)} \ , \tag{4.2}$$

**Algorithm 2** Pref-CBR Search($K$, $J$)

**Require:** $K$ = number of nearest neighbors collected in the case base
$J$ = number of preferences used to guide the search process

$\mathbb{X}_0 \leftarrow$ list of problems to be solved ▷ a subset of $\boldsymbol{X}$
$Q \leftarrow [\cdot]$ ▷ empty list of performance degrees
**CB** $\leftarrow \emptyset$ ▷ initialize empty case base
**while** $\mathbb{X}_0$ not empty **do**
  $\boldsymbol{x}_0 \leftarrow$ pop first element from $\mathbb{X}_0$ ▷ new problem to be solved
  $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\} \leftarrow$ nearest neighbors of $\boldsymbol{x}_0$ in **CB** (according to $\Delta_X$)
  $\{\mathcal{P}(\boldsymbol{x}_1), \ldots, \mathcal{P}(\boldsymbol{x}_K)\} \leftarrow$ preferences associated with nearest neighbors
  $\mathcal{P} \leftarrow \mathcal{P}(\boldsymbol{x}_1) \cup \mathcal{P}(\boldsymbol{x}_2) \cup \ldots \cup \mathcal{P}(\boldsymbol{x}_k)$ ▷ combine neighbor preferences
  $\boldsymbol{y}^\bullet \leftarrow \mathrm{CBI}(\mathcal{P}, \mathbb{Y})$ ▷ select an initial candidate solution
  $\mathbb{Y}^{vis} \leftarrow \{\boldsymbol{y}^\bullet\}$ ▷ candidates already visited
  $\mathcal{P}_0 \leftarrow \emptyset$ ▷ initialize new preferences
  **repeat**
    $\mathcal{P}^{nn} = \{\boldsymbol{y}^{(j)} \succ \boldsymbol{z}^{(j)}\}_{j=1}^{J} \leftarrow J$ preferences in $\mathcal{P} \cup \mathcal{P}_0$ closest to $\boldsymbol{y}^\bullet$
    $\mathbb{Y}^{nn} \leftarrow$ neighborhood $\mathcal{N}(\boldsymbol{y}^\bullet)$ of $\boldsymbol{y}^\bullet$ in $\mathbb{Y} \setminus \mathbb{Y}^{vis}$
    $\boldsymbol{y}^{query} \leftarrow \mathrm{CBI}(\mathcal{P}^{nn}, \mathbb{Y}^{nn})$ ▷ find next candidate
    $[\boldsymbol{y} \succ \boldsymbol{z}] \leftarrow \mathrm{Oracle}(\boldsymbol{x}_0, \boldsymbol{y}^{query}, \boldsymbol{y}^\bullet)$ ▷ check if new candidate is better
    $\mathcal{P}_0 \leftarrow \mathcal{P}_0 \cup \{\boldsymbol{y} \succ \boldsymbol{z}\}$ ▷ memorize preference
    $\boldsymbol{y}^\bullet \leftarrow \boldsymbol{y}$ ▷ adopt the current best solution
    $\mathbb{Y}^{vis} \leftarrow \mathbb{Y}^{vis} \cup \{\boldsymbol{y}^{query}\}$
  **until** convergence
  $q \leftarrow$ performance of solution $\boldsymbol{y}^\bullet$ for problem $\boldsymbol{x}_0$
  $Q \leftarrow [Q, q]$ ▷ store the performance
  **CB** $\leftarrow$ **CB** $\cup \{(\boldsymbol{x}_0, \mathcal{P}_0)\}$ ▷ memorize new experience
  Adapt distance measure $\Delta_Y$
**end while**
return list $Q$ of performance degrees

where $S_Y(\boldsymbol{y}, \boldsymbol{y}^*)$, which is defined as

$$S_Y(\boldsymbol{y}, \boldsymbol{y}^*) \;=\; \exp\big(-\beta \cdot \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)\big)$$

according to (4.1), can be seen as the degree to which $\boldsymbol{y}$ resembles the ideal solution $\boldsymbol{y}^*$; likewise, $S_Y(\boldsymbol{z}, \boldsymbol{y}^*)$ is the degree to which $\boldsymbol{z}$ is close to ideal. Thus, the probability of observing the (revealed) preference $\boldsymbol{y} \succ \boldsymbol{z}$ depends on the degree of optimality of $\boldsymbol{y}$ and $\boldsymbol{z}$, namely their respective closeness to the ideal solution: The less optimal $\boldsymbol{z}$ in comparison to $\boldsymbol{y}$, the larger the probability to observe $\boldsymbol{y} \succ \boldsymbol{z}$; if $\Delta_Y(\boldsymbol{z}, \boldsymbol{y}^*) = \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)$, then $\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z}) = 1/2$.

### 4.1.3 Distance learning of the solutions in Pref-CBR

This section is devoted to the main extension of our Pref-CBR framework, namely the distance adaptation component in line 24 of Algorithm 2. In our framework, we assume preference information to be produced according to the probabilistic model (4.2). Therefore, it is natural to approach the distance learning problem from a probabilistic point of view. Correspondingly, we shall propose a Bayesian method to tackle this problem.

#### 4.1.3.1 A local-global representation of distance

We begin with a simplifying assumption on the structure of the distance measure $\Delta_Y$, namely that it adheres to the *local-global principle* [87] and takes the form

$$\Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*) = \sum_{i=1}^{k} \alpha_i \cdot \Delta_i(\boldsymbol{y}, \boldsymbol{y}^*) \;, \tag{4.3}$$

where $\Delta_1, \ldots, \Delta_k$ are local distances pertaining to different properties of solutions, and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$ is a partition of unity (i.e., the coefficients $\alpha_i$ are non-negative and sum up to 1). We assume the $\Delta_i$ to be known, whereas the $\alpha_i$, which are modeling the importance of the local distances, are supposed to be unknown. Learning the distance measure (4.3) is thus equivalent to learning these parameters.

#### 4.1.3.2 Bayesian distance learning

Adopting the above representation of the distance measure $\Delta_Y$, our choice model (4.2) is now given by

$$\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z}) \;=\; \frac{S_Y(\boldsymbol{y}, \boldsymbol{y}^*)}{S_Y(\boldsymbol{y}, \boldsymbol{y}^*) + S_Y(\boldsymbol{z}, \boldsymbol{y}^*)} \tag{4.4}$$

with

$$S_Y(\boldsymbol{y}, \boldsymbol{y}^*) = \exp\left( -\sum_{i=1}^{k} \gamma_i \cdot \Delta_i(\boldsymbol{y}, \boldsymbol{y}^*) \right) \tag{4.5}$$

and $\gamma_i = \beta \cdot \alpha_i \geq 0$. Thus, learning $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_k)$ means learning $\beta$ and $\alpha$ simultaneously. In fact, these parameters can be recovered from $\boldsymbol{\gamma}$ as follows:

$$\begin{aligned} \beta &= \gamma_1 + \gamma_2 + \ldots + \gamma_k \\ \alpha_i &= \gamma_i / \beta \end{aligned}$$

For simplicity, suppose that $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_k)$ only assumes to take values from a finite (or at least countable) set $\Gamma \subset \mathbb{R}_+^k$; this allows us to work with probability distributions instead of density functions. For example, $\Gamma$ could be a suitable discretization of a continuous domain, such as a grid on a hypercube.

Since the true $\boldsymbol{\gamma}$ (used by the oracle) is assumed to be unknown, we model our belief about the parameters $\gamma_i$ in (4.5) in the form of a probability distribution

$$\mathbf{P} : \Gamma \to [0, 1] \;\; ,$$

i.e., for each vector $\boldsymbol{\gamma} \in \Gamma$, $\mathbf{P}(\boldsymbol{\gamma})$ denotes the prior probability of that vector. Unless specific (prior) knowledge is available, this probability can be initialized by the uniform distribution over $\Gamma$.

Now, suppose a preference $p = [\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^*]$ to be revealed by the oracle, recalling that $\boldsymbol{y}^*$ is unknown and that each preference given by the oracle hints at it. Since the oracle is supposed to generate preferences according to (4.4), this observation provides a hint at the true value of $\boldsymbol{\gamma}$. More specifically, it can be used for performing a Bayesian inference step to update our belief about $\boldsymbol{\gamma}$:

$$\mathbf{P}(\boldsymbol{\gamma} \,|\, p) = \frac{\mathbf{P}(p \,|\, \boldsymbol{\gamma}) \mathbf{P}(\boldsymbol{\gamma})}{\mathbf{P}(p)} \;\; , \tag{4.6}$$

where $\mathbf{P}(p \,|\, \boldsymbol{\gamma})$ is given by (4.4). Concretely, this means realizing the following update for each $\boldsymbol{\gamma} \in \Gamma$:

$$\mathbf{P}(\boldsymbol{\gamma}) \;\leftarrow\; \frac{1}{C} \cdot \mathbf{P}(\boldsymbol{\gamma}) \cdot \mathbf{P}(p \,|\, \boldsymbol{\gamma}) \;\; ,$$

where $\mathbf{P}(p \mid \boldsymbol{\gamma}) = \mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z})$ is given by (4.4) and $C$ is a normalizing constant assuring that the (posterior) probability degrees sum up to 1. To the best of our knowledge, there is no parametrized family of distributions that is conjugate with (4.4), so that the posterior (4.6) needs to be computed numerically.

#### 4.1.3.3 Integration with Pref-CBR search

As mentioned before, the adaptation of $\Delta_Y$ as outlined above is integrated in our Pref-CBR search procedure in line 24 of Algorithm 2. Thus, the idea is to update the belief about $\boldsymbol{\gamma}$ (and hence about $\Delta_Y$, which is uniquely determined by this parameter) after each problem solving episode, making use of the newly observed preferences. Here is a summary of the main steps:

- Suppose our current belief about $\boldsymbol{\gamma}$ to be specified in the form of a probability $\mathbf{P}$ on $\Gamma$; in the beginning, this could be the uniform distribution, for example.

- In a single problem solving episode (lines 5–23 of Algorithm 2), Pref-CBR Search is used to solve a new problem $\boldsymbol{x}_0$. This requires a concrete distance $\Delta_Y$, and therefore a concrete parameter vector $\boldsymbol{\gamma}$, which is used to "mimic" the (ground-truth) similarity measure of the oracle. To this end, we can reasonably choose the expectation according to our current distribution, which is considered as our current "best guess" of the true vector:[1]

$$\widehat{\boldsymbol{\gamma}} = \frac{1}{|\Gamma|} \sum_{\boldsymbol{\gamma} \in \Gamma} \boldsymbol{\gamma} \cdot \mathbf{P}(\boldsymbol{\gamma}) \tag{4.7}$$

  Using the distance measure (4.3) and choice model (4.4) parameterized by $\widehat{\boldsymbol{\gamma}}$ or, more specifically, the induced parameters

$$\widehat{\beta} = \widehat{\gamma}_1 + \ldots + \widehat{\gamma}_k \ , \tag{4.8}$$

$$\widehat{\alpha}_i = \widehat{\gamma}_i / \widehat{\beta} \ , \tag{4.9}$$

  the Pref-CBR search procedure is performed as usual.

- Upon termination of a problem solving episode (line 20 of Algorithm 2), Pref-CBR Search yields a solution $\boldsymbol{y}^\bullet$, which is not necessarily the truly ideal solution

[1] An alternative would be to choose the mode of the distribution instead of the mean.

$\boldsymbol{y}^*$ but at least an approximation thereof. Moreover, Pref-CBR Search returns a set of preferences $\mathcal{P}_0$ that have been collected during the search process. These preferences can now be used for updating our belief about $\boldsymbol{\gamma}$.[1] To this end, the adaptation (4.6) is carried out for each of the preferences in $\mathcal{P}_0$. More specifically, for each preference $\boldsymbol{y} \succ \boldsymbol{z}$ observed in the last episode, a learning step is carried out with $[\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^\bullet]$.

It is important to note that contextualizing an observed preference $\boldsymbol{y} \succ \boldsymbol{z}$ by $\boldsymbol{y}^\bullet$ instead of $\boldsymbol{y}^*$ makes our distance learning method *approximate* and may possibly affect its efficacy. In fact, since preferences of the form $\widehat{p} = [\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^\bullet]$ can be seen as "noisy" versions of the true preferences $p = [\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^*]$, our method is actually learning from noisy data. We shall return to this issue in the experimental subsection further below.

### 4.1.4 Synthetic data illustration

To illustrate our Bayesian approach to distance learning, independently of its use within the Pref-CBR framework, we conducted some very simple experiments for the case $\mathbb{Y} = [0,1]^2$, $\Delta_1(\boldsymbol{y}, \boldsymbol{y}') = |y_1 - y_1'|$, $\Delta_2(\boldsymbol{y}, \boldsymbol{y}') = |y_2 - y_2'|$, and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$. For simplicity, we also assumed $\beta$ to be known and only learned $\boldsymbol{\alpha}$.

To this end, we generated triplets $(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{y}^*) \subset \mathbb{Y}$ uniformly at random and derived exemplary preferences $[\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^*]$ or $[\boldsymbol{z} \succ \boldsymbol{y} \,|\, \boldsymbol{y}^*]$ according to our probabilistic model (4.2). Starting with a uniform prior on the simplex $\{(\alpha_1, \alpha_2) \,|\, \alpha_1, \alpha_2 \geq 0,\, \alpha_1 + \alpha_2 = 1\}$, $N$ updates (4.6) were realized based on $N$ random preferences of that kind.

Figure 4.1 shows typical examples of the marginal distributions for $\alpha_1$ after $N = 50$, $N = 200$ and $N = 500$ examples. As expected, the distributions are fluctuating around the true value of $\alpha_1$ (here taken as 0.3) and become more and more peaked with increasing $N$. Moreover, comparing the distributions on the left and the right panel, it can be seen that learning becomes easier for larger values of $\beta$: for $\beta = 5$, the distributions are less peaked than for $\beta = 10$. Since $\beta$ reflects the reliability of the oracle, this is again in agreement with our expectation. The same effect can also be observed in Figure 4.2 (left), which shows the boxplots for the mean value estimator (4.9); 100 of such estimators were derived from distributions for $N = 100$ and different values of

[1] In principle, this set of preferences can be enriched further, assuming that each solution adopted in a later stage of the search process is preferred to each earlier solution.

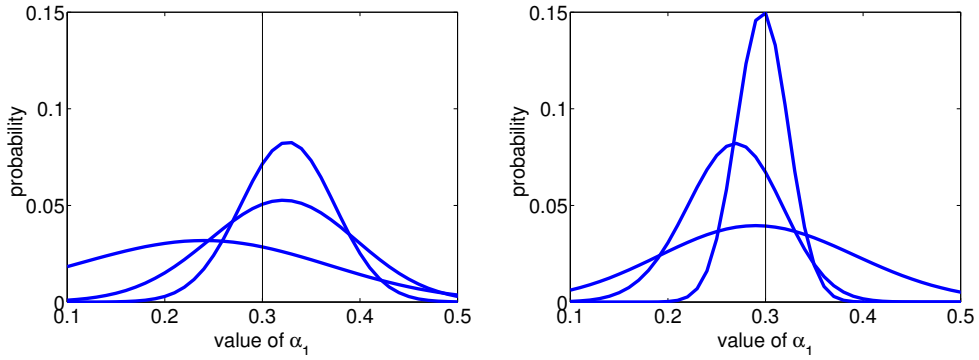


**Figure 4.1:** Probability distributions for the parameter $\alpha_1$ after $N = 50$, $N = 200$ and $N = 500$ examples, with $\beta = 5$ (left) and $\beta = 10$ (right).

$\beta$. Again, the larger $\beta$, the more precise the estimate of $\alpha_1$ (and correspondingly of $\alpha_2$).

As explained above, our Pref-CBR framework only produces an estimate $\boldsymbol{y}^\bullet$ of the ideal solution $\boldsymbol{y}^*$. Therefore, our distance learning method is based on preferences $[\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^\bullet]$ that can be seen as "noisy" versions of the true preferences $[\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^*]$. To simulate this property, we generated triplets $(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{y}^*) \subset \mathbb{Y}$ as above and set $\boldsymbol{y}^\bullet = \boldsymbol{y}^* + (\epsilon_1, \epsilon_2)^\top$, where $\epsilon_1$ and $\epsilon_2$ are normally distributed random variables with mean 0 and standard deviation $\sigma$. The observed preference (either $\boldsymbol{y} \succ \boldsymbol{z}$ or $\boldsymbol{z} \succ \boldsymbol{y}$) was then generated with our model (4.2) using the true $\boldsymbol{y}^*$, while distance learning was done using this model with the estimate $\boldsymbol{y}^\bullet$.

The effect of learning from noisy examples can be seen in Figure 4.2 (right), where we again show boxplots for the mean value estimator (4.9) based on 100 repetitions of the learning procedure with $N = 100$. As can be seen, the noise level $\sigma$ does not seem to have a strong influence on the *variance* of the estimation. What is notable, however, is an apparent *bias* of the estimate: The larger $\sigma$, the more the estimates of $\alpha_1$ are moving away from the true value 0.3 toward 0.5. Although this result cannot easily be generalized beyond the specific setting of our experiment, a tendency toward uniform weights of the $\alpha$-coefficients (i.e., $\alpha_1 = \alpha_2 = 0.5$ in our case) is plausible: The more $\boldsymbol{y}^\bullet$ deviates from $\boldsymbol{y}^*$, the more noisy the examples will be for our distance learner—in the limit, they will become purely random, and on average, all local distances $\Delta_i$ will seemingly have the same influence then.

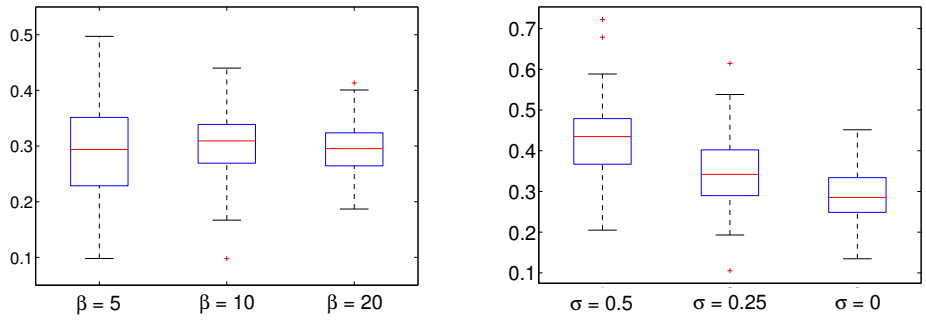


**Figure 4.2:** Boxplots for the mean value estimate of $\alpha_1$. Left: Different values of the precision parameter $\beta$. Right: Different levels of noise in the estimate $\boldsymbol{y}^{\bullet}$.

## 4.2 Learning similarity measures in the problem space

To further improve search performance of our Pref-CBR framework, we extend the learning of similarity measures also for the problem space. Our inference method which we use in our framework, allowed us to have a sound theoretical basis on which we used the Bayesian method for learning similarity measures in the solution space; in the problem space we do not have this theoretical basis. Therefore, we propose another learning method for our problem space, which learns how to combine given local similarity measures into a global one, and breaks down the learning problem into a binary classification problem. We choose to use the perceptron algorithm for the learning, where positive and negative examples for the perceptron learning are created in a way to fit and integrate with our framework. It is to be noted that we use a "noise-tolerant", robust version of the perceptron in our framework, the SoftDoubleMaxMinOver perceptron [88].

### 4.2.1 Related work

The learning of similarity measures of the problems has been tackled by previous work, some of which we will briefly recall. Learning weights in a linear combination of local similarity functions has been considered in the work of [82]. Stahl and Gabel in [85] also address learning *local* similarity measures by proposing evolutionary optimization techniques for adaptation. Feature weighing and selection methods using $k$-NN classification have also been addressed by other authors [89, 90, 91]. Earlier work on

comparing feature weighting methods has been also discussed by [92]. Feature weighing methods in machine learning have also been proposed by the research of [93, 94].

The work which resembles most our proposed method, is the one discussed by [17]. The authors use machine learning methods to elicit global similarity measures on the basis of feedback in the form of examples (preferences). In the former work and in ours, we use qualitative feedback to create the examples for learning: given a reference case and two cases to compare with, we use qualitative information about which of these two cases is more similar to the reference case. Essentially, this is what Stahl in [70] refers to as *relative case utility feedback*.

### 4.2.2 Distance learning of the problems in Pref-CBR

Following our assumption, the problem space $\mathbb{X}$ is equipped with a similarity measure $S_X : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_+$, to be linked to a (reciprocal) distance measure $\Delta_X : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_+$. Thus, for any pair of problems $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{X}$, their similarity is denoted by $S_X(\boldsymbol{x}, \boldsymbol{x}')$ and their distance by $\Delta_X(\boldsymbol{x}, \boldsymbol{x}')$. We will propose the perceptron algorithm for learning by examples.

#### 4.2.2.1 A local-global representation of distance

We assume the existence of $d$ *local distance measures*

$$\Delta_X : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_+ \quad (i = 1 \dots d). \tag{4.10}$$

For each pair of cases $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{X}$, $\Delta_X(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}_+$ is a measure of the distance between these cases with respect to a certain aspect.

According to the *local–global principle*, the (global) distance between two cases can be obtained as an aggregation of the local distance measures (4.10):

$$\Delta_X(\boldsymbol{x}, \boldsymbol{x}') \;=\; \mathrm{AGO}\left(\, \Delta_{X1}(\boldsymbol{x}, \boldsymbol{x}'), \Delta_{X2}(\boldsymbol{x}, \boldsymbol{x}') \dots \Delta_{Xd}(\boldsymbol{x}, \boldsymbol{x}') \,\right), \tag{4.11}$$

where AGO is a suitable aggregation operator. As a special case, consider a representation of cases in terms of $d$-dimensional feature vectors

$$\boldsymbol{x} = (x_1, x_2 \dots x_d) \in \mathbb{X}_1 \times \mathbb{X}_2 \times \dots \times \mathbb{X}_d, \tag{4.12}$$

where $\mathbb{X}_i$ is the domain of the $i$-th attribute $X_i$. $\mathbb{X}$ is then given by the Cartesian product of these domains, $\mathbb{X}_1 \times \mathbb{X}_2 \times \ldots \times \mathbb{X}_d$, and the local distances are of the form

$$\Delta_{X\,i} : \mathbb{X}_i \times \mathbb{X}_i \to \mathbb{R}_+, \tag{4.13}$$

i.e., $\Delta_{X\,i}(x_i, x_i')$ assigns a distance to each pair of attributes $(x_i, x_i') \in \mathbb{X}_i \times \mathbb{X}_i$; obviously, (4.13) is a special case of (4.10).

#### 4.2.2.2 Perceptron distance learning

In our work, we use a simple aggregation operator which is a linear combination:

$$\Delta_X(\boldsymbol{x}, \boldsymbol{x}') \;=\; \sum_{i=1}^{d} w_i \cdot \Delta_{X\,i}(\boldsymbol{x}, \boldsymbol{x}'). \tag{4.14}$$

For model interpretation we require non-negative weights

$$\boldsymbol{w} = (w_1 \ldots w_d) \;\geq\; 0 \tag{4.15}$$

in order to guarantee the monotonicity of the distance measure (4.11). That is, if a local distance increases, the global distance cannot decrease.

In addition to the simplicity of the linear model (4.14), its interpretation of a weight $w_i$ corresponds directly to the *importance* of a local measure, where $\sum_{i=1}^{d} w_i = 1$. In principle, it thus also allows one to incorporate additional background knowledge in a convenient way, e.g., that attribute $x_i$ is at least as important as attribute $x_j$ ($w_i \geq w_j$). Finally, the linear model is attractive from a machine learning point of view, as it is considered to be easily determined by learning algorithms.

For optimal performance of any CBR system, there exists the need for efficiently choosing nearest neighbors. The learned weights directly have an influence on the choice of cases during the $k$ nearest neighbors' retrieval, thus leading to an improved performance. Accordingly, the training information to be given as input to our learner is of the following form: case $\boldsymbol{x_0}$ is more similar to $\boldsymbol{x_1}$ than to $\boldsymbol{x_2}$. Given a set of training data for the learner, in the form of: case $\boldsymbol{x_0}$ is more similar to case $\boldsymbol{x_1}$ than to case $\boldsymbol{x_2}$, we start to define our learning problem. The basic learning problem is to find a distance function (4.14), which is as much as possible in agreement with these constraints and also satisfies the monotonicity property (4.15). We reduce the above learning problem to a *binary classification problem.*

Due to the assumption of a linear distance model, it is then plausible to state: The inequality $\Delta_X(\boldsymbol{x_0}, \boldsymbol{x_1}) < \Delta(\boldsymbol{x_0}, \boldsymbol{x_2})$ required by a constraint $(\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2})$ is equivalent to

$$\langle \boldsymbol{w}, \boldsymbol{t} \rangle = \sum_{i=1}^{d} w_i \cdot t_i > 0,$$

where $t_i \stackrel{\text{df}}{=} \Delta_{X\,i}(\boldsymbol{x_0}, \boldsymbol{x_2}) - \Delta_{X\,i}(\boldsymbol{x_0}, \boldsymbol{x_1})$. From a classification point of view, $\boldsymbol{t} = T(\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2}) = (t_1 \ldots t_d)$ is hence a positive example and $-\boldsymbol{t}$ a negative one. That is, a similarity constraint $(\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2})$ can be transformed into two examples $(\boldsymbol{t}, +1)$ and $(-\boldsymbol{t}, -1)$ for binary classification learning.

Moreover, the vector $\boldsymbol{t} = (t_1 \ldots t_d)$ defines the distance function (4.14). It is noteworthy to say that the well-known perceptron algorithm is an error-driven on-line algorithm that adapts the weight vector $\boldsymbol{w}$ in an incremental way. To guarantee monotonicity, we simply modify this algorithm as follows: Each time an adaptation of $\boldsymbol{w}$ produces a negative component $\boldsymbol{w_i} < 0$, this component is set to 0. In this way, the original adaptation is replaced by a "thresholded" adaptation.

In its basic form, the perceptron algorithm provably converges after a finite number of iterations, provided the data is linearly separable. As we mentioned, the specific perceptron we use in our framework, from [88], provides maximum margin classifier and converges without a bias and thus yields more reliable weights. A pseudo-code of the DoubleMaxMinOver perceptron algorithm that we used is shown in Figure 4.3, where $\boldsymbol{w}^T \boldsymbol{x}_i$ is the weight vector transposed times the i-th instance vector of $\boldsymbol{x}$. The result of this product is a simple number; if it is negative, the instance is at the *left* side of the decision boundary (described by weight vector $\boldsymbol{w}$), and if the product is positive then it is on the *right* side of the boundary. Accordingly, if the weight is properly chosen, all positive instance categories are on the positive side and all negative categories are on the negative side. The algorithm shown in Figure 4.3 adjusts the classification boundary $\boldsymbol{w}$ in such a way as to maximize the margin to both class instances even if data is not linearly separable by such a classification boundary, and it converges after a finite number of iterations.

#### 4.2.2.3 Integration with Pref-CBR search

The adaptation of $\Delta_X$ as outlined above, is integrated in our Pref-CBR search procedure directly following line 24 of Algorithm 2. The adaptation of $\boldsymbol{w}$ (and hence about $\Delta_X$,

**Algorithm 3** DoubleMaxMinOver (for $t = 0$ one chooses $\mathbf{x}_{\min^+} = \mathbf{x}_{\max^+}$ and $\mathbf{x}_{\min^-} = \mathbf{x}_{\max^-}$):

$\mathbf{w} \leftarrow 0$
**for** $t = 0$ to $t_{\max}$ **do**
  $\mathbf{x}_{\min^+} \leftarrow \arg \min_{\mathbf{x}_i \in \mathcal{X}^+} \left(\mathbf{w}^T \mathbf{x}_i\right)$
  $\mathbf{x}_{\min^-} \leftarrow \arg \min_{\mathbf{x}_i \in \mathcal{X}^-} \left(-\mathbf{w}^T \mathbf{x}_i\right)$
  $\mathbf{x}_{\max^+} \leftarrow \arg \max_{\mathbf{x}_i \in \mathcal{V}_t^+} \left(\mathbf{w}^T \mathbf{x}_i\right)$
  $\mathbf{x}_{\max^-} \leftarrow \arg \max_{\mathbf{x}_i \in \mathcal{V}_t^-} \left(-\mathbf{w}^T \mathbf{x}_i\right)$
  $\mathbf{w} \leftarrow \mathbf{w} + 2(\mathbf{x}_{\min^+} - \mathbf{x}_{\min^-}) - (\mathbf{x}_{\max^+} - \mathbf{x}_{\max^-})$

**Figure 4.3:** SoftDoubleMaxMinOver perceptron algorithm

which is uniquely determined by this parameter) occurs after each problem solving episode. Here is a summary of the main steps:

- In the beginning we set equal weights $\boldsymbol{w} = (\frac{1}{d}, ..., \frac{1}{d})$, for a $d$-dimensional feature vector of a problem $\boldsymbol{x}$.

- In a single problem solving episode (lines 5–23 of Algorithm 2), Pref-CBR Search is used to solve a new problem $\boldsymbol{x}_0$. This requires a concrete distance $\Delta_X$, and therefore a concrete parameter vector $\boldsymbol{w}$, which is used for $k$ NN retrieval. This parameter vector is an estimate of the learning, given previously solved problems. The current problem is the reference, and it is compared with two previously solved problems for obtaining similarity information for the perceptron learner.

- Following the CBR assumption and our Pref-CBR framework, we check our solved problems in the case base with our current case, to form our examples for the perceptron learner. Using the distance measure (4.14) and the inequality

$$\langle \boldsymbol{w}, \boldsymbol{t} \rangle = \sum_{i=1}^{d} w_i \cdot t_i > 0,$$

  where $t_i \stackrel{\text{df}}{=} \Delta_{Xi}(\boldsymbol{x_0}, \boldsymbol{x_2}) - \Delta_{Xi}(\boldsymbol{x_0}, \boldsymbol{x_1})$. Thus in addition to $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$, we can form our perceptron training examples from the rest of the solved cases accordingly.

- The class of the collected training examples is assigned by the following procedure:

  Referring to the likelihood function in Chapter 3, we compare the probability of the preferences $\mathcal{P}_1$ with the probability of the preferences $\mathcal{P}_2$ under associated (ML) parameters $(\boldsymbol{y}_0^\bullet, \beta)$.

  If distances of problems

  $$\Delta_X(\boldsymbol{x_0}, \boldsymbol{x_1}) < \Delta_X(\boldsymbol{x_0}, \boldsymbol{x_2}) \tag{4.16}$$

  which means if the case $\boldsymbol{x_1}$ is closer (more similar) to the case $\boldsymbol{x_0}$ than the case $\boldsymbol{x_2}$, and these distances are in accordance with the likelihood of their preferred solutions

  $$\ell(\boldsymbol{y}_0^\bullet, \beta \,|\, \mathcal{P}_1) > \ell(\boldsymbol{y}_0^\bullet, \beta \,|\, \mathcal{P}_2) \tag{4.17}$$

  where $\mathcal{P}_1$ is the set of preferences over solutions of $\boldsymbol{x_1}$ and $\mathcal{P}_2$ the set of preferences over solutions of $\boldsymbol{x_2}$ and $\boldsymbol{y_0}$ is the best found solution of $\mathcal{P}_0$, then

  $$t_i \stackrel{\text{df}}{=} \Delta_{Xi}(\boldsymbol{x_0}, \boldsymbol{x_2}) - \Delta_{Xi}(\boldsymbol{x_0}, \boldsymbol{x_1}) \tag{4.18}$$

  is a positive example, else $t_i$ is a negative example. To further explain the generation of examples let us note that for one case to be closer to the current case than a third case should indicate that the solution of the closer case, thus its preferences over solutions, should also be closer to the preference over solutions of the current case than the third case. In other words, if we check for how likely we obtain the final solution $\boldsymbol{y_0}$ for case $\boldsymbol{x_0}$ from the generated preferences of the closer case $\boldsymbol{x_1}$, we should be getting a higher likelihood than from the generated preferences of the farther away case $\boldsymbol{x_2}$. If that is the case then we can consider that 4.18 is a positive example. If we get a higher likelihood for solution $\boldsymbol{y_0}$ from the generated preferences of the case $\boldsymbol{x_2}$, then we consider 4.18 to be a negative example.

- Upon termination of line 23 of Algorithm 2, and after collecting the formed examples for the perceptron learner and running the perceptron, we adapt the distance measure $\Delta_X$. The Pref-CBR search procedure is performed as usual for the new problem, using the updated parameter vector $\boldsymbol{w}$ for $k$ NN retrieval.

## 4.3 Conclusion

In this chapter we proposed two methods for learning similarity measures: the Bayesian approach for the solution space metrics' learning and the perceptron algorithm for the problem space metrics' learning. We described in detail how each approach is implemented and integrated within our Pref-CBR problem-solving framework. These learning measures improve the performance of the problem solving process and illustrations showing the efficacy of the explained methods are shown in Chapter 7.

# 5

# Case Base Maintenance in Pref-CBR

In preference-based CBR (Pref-CBR), as we have stated in Chapter 3, problem solving experience is represented in the form of contextualized preferences, namely, preferences between candidate solutions in the context of a target problem to be solved. In each step, the current best solution $\boldsymbol{y}$ is compared with another, slightly modified/adapted solution $\boldsymbol{z}$, and the better one is retained. Since a single comparison is assumed to be costly, the number of adaptation steps is limited. Nevertheless, each step gives rise to a piece of information $\boldsymbol{y} \succ_{\boldsymbol{x}} \boldsymbol{z}$. Therefore, a single *case* eventually consists of a problem $\boldsymbol{x}$ together with a set of (pairwise) preferences over solutions (instead of merely a single solution, like in conventional CBR). Since a potentially large number of such preferences can be collected during the course of each problem solving episode, case base maintenance clearly becomes an issue in Pref-CBR.

It is clear that simply storing each encountered problem along with a set of associated preferences is not advisable, especially since a case base of that type may quickly become too large and hamper efficient case retrieval; besides, many of the preferences collected in a problem solving episode will be redundant to some extent. In CBR, this problem has been addressed by methods for *case base maintenance* [19]. Such methods seek to maintain the problem solving competence of a case base by applying some *case base editing* strategies, including the removal of misleading (noisy) or redundant cases. Case base maintenance (CBM) proved essential to guarantee the efficiency and performance of CBR systems. According to the aforesaid about preferences being collected

in the case base over time, it might be even more critical for preference-based than for conventional CBR to apply case base maintenance.

In this chapter, we therefore address the problem of case base maintenance and extend our Pref-CBR framework by another component, namely, a method for dynamic case base maintenance. This method consists of four strategies, two of the strategies come under whole case deletion, and the other two under partial case deletion from the case base. The main goal of these strategies is to increase efficiency of case-based problem solving, by reducing the size of the case base, while maintaining its performance. Despite being inspired by existing CBM techniques for conventional CBR, our strategies are specifically tailored to our framework and exploit properties of the underlying preference-based representation of problem solving experience.

## 5.1 Related work

The growing demand for case-base maintenance of CBR systems has led to intensive research on examining different aspects concentrating on maintaining the case base, while retaining its competence [95, 96]. This is also our goal for the strategies we propose in this chapter. There are several approaches to case base maintenance: focusing on choosing noisy or redundant cases based on their utility degree, competence contributions, their effect on overall performance or on their influence of providing adaptation for new problems to be solved. The work mentioned below provides briefly some information about the different methods.

Several CBR methods implement strategies that focus on choosing which noisy or redundant cases to delete from the case base. The simplest strategy is random deletion, which is initiated once a given limit of the size of the case base is exceeded [97]; obviously, this method guarantees a bound on the size but no preservation of the competence of the case base. A more principled approach is utility deletion, where the utility of a case is measured by its performance benefits (e.g., given by Minton's utility metric) [98]; cases with negative utility are removed. There are other methods such as footprint deletion and footprint utility deletion, which specify the cases to be deleted based on their competence contributions [96]. The cases are categorized into pivotal, spanning, support and auxiliary; pivotal cases have highest effect on competence, while auxiliary cases have lowest effect [99]. Modifying the idea of coverage (the set of target

problems a case can solve) and reachability (the set of cases that can provide a solution for a target problem) of a case as introduced in [96]. Cases are identified by their coverage and reachability values based on rough set theory for categorizing data in [100], and accordingly relevance of each case is extracted.

Other maintenance methods focus more on an increase in efficiency, in terms of memory storage size and computation time of solving problems [101]. This increase in efficiency could in return cause some degradation in performance. One well-known method is based on the condensed nearest neighbor (CNN) rule by [102], where a subset of the case base is selected, which should perform almost as well as the original case base in classifying new cases. CNN was then extended by selective nearest neighbor (SNN); any case in the original case base must be closer to a case in the formed subset belonging to the same class, than to any case in the original case base belonging to a different class [103]. Reduced edited nearest neighbor (RENN) method further extends CNN by removing noisy cases, which have a different class than the majority of their nearest neighbors; it is computationally more expensive than CNN [104] though. Also described in [104], the blame based noise reduction (BBNR) method deletes cases that cause other cases to be misclassified. A case base can also be reduced as explained by [105], where a subset of the case base is formed in which selection of cases is based on some "justifications". These justifications are being output from using a (lazy) machine learning method; this selection procedure resembles the competence selection of cases in [97], but in the former the selection of cases is based on the justifications rather than the competence.

Additionally, other maintenance methods called adaptation-guided case base maintenance methods, base the selection of cases to be retained in the case base on both their value in solving problems and on their value in generating new adaptation rules; these adaptation rules contribute to the knowledge for later problem solving [19]. Complexity-informed maintenance is another method presented in [106]; it provides redundancy reduction and offers a compromise between a smaller case base and greater accuracy. Case complexity enables varying levels of aggressiveness in redundancy and error reduction maintenance algorithms, thus compromising between amount of reduction and correspondingly level of performance. The higher the aggressiveness, the more reduction in case base size and correspondingly the lower the performance level.

The previously listed methods are used to either increase the efficiency of the case-based reasoning system while maintaining its competence, or having a trade-off between an increase in the level of efficiency and a decrease in the level of performance. The case base is maintained when a certain size limit is reached, or by setting periodic time slots for the maintenance to be performed. As pointed out by [107], to tackle performance problems of a CBR system, the goal would be to update the existing case base while maintaining problem solving competence. This is also the goal of our maintenance strategies, which are specifically designed for the Pref-CBR problem solving framework.

## 5.2 Case base maintenance for Pref-CBR

Most methods for case base maintenance make use of two important criteria for case addition or removal, namely, noise and redundancy. A "noisy" case is a case that differs significantly from its (nearby) neighbors and, therefore, violates the regularity assumption underlying CBR. Retrieving such a case and using it to solve a new problem should obviously be avoided, whence it should better not be stored in the case base. A redundant case, on the other side, is very similar to its neighbors and, therefore, does hardly provide additional information, at least if enough other cases have already been stored. Such cases can often be removed to reduce the size of the case base without compromising performance.

In Pref-CBR, a case does not only contain a single solution, like in conventional CBR, but rather a set of preferences. Thus, instead of either retaining or removing a complete case, there is in principle the possibility to retain or remove a *part* of a case, simply by retaining or removing a part of the pairwise preferences. In fact, as will be seen later on, both noise and redundancy can occur on the level of a single case as well as on the level of the case base.

First of all, however, one should clarify what noise and redundancy may actually refer to in the context of Pref-CBR. In fact, it is important to note that a piece of information is not noisy or redundant per se. First, it can only be noisy or redundant when being considered jointly with other information. Moreover, what also needs to be taken into consideration is the way in which the information will be (re-)used: What is the influence of the information on future problem solving episodes?

### 5.2.1 Noise and redundancy in Pref-CBR

To answer this question, recall the key idea and basic inference principle of Pref-CBR: An observed preference $\boldsymbol{y} \succ \boldsymbol{z}$ provides a kind of "directional hint" in the solution space $\mathbb{Y}$. It suggests moving toward those solutions $\boldsymbol{y}^*$ for which the probability

$$\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^*) \;=\; \frac{1}{1 + \exp\Big( -\beta\big(\Delta_Y(\boldsymbol{z}, \boldsymbol{y}^*) - \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)\big)\Big)} \tag{5.1}$$

is large, hence making $\boldsymbol{y}^*$ likely as a solution for the problem at hand, and away from those solutions for which the probability of observing this preference is small. Likewise, a whole set of preferences $\mathcal{P}$ suggest moving toward those solutions for which the combined likelihood

$$\ell(\theta) = \ell(\theta \,|\, \mathcal{D}) = \prod_{i=1}^{N} \mathbf{P}\left(\boldsymbol{y}^{(i)} \succ \boldsymbol{z}^{(i)} \,|\, \theta\right) \tag{5.2}$$

is large, and away from those solutions for which this likelihood is small. Roughly speaking, the likelihood function combines the individual hints into a single one.

Now, we propose the following distinction between noise and redundancy on the level of a single case and the level of the case base.

- **Intra-case redundancy**: Pairwise comparisons collected during a problem solving episode can obviously be redundant to some extent, in particular because the same solutions will be shared among many of these comparisons. Moreover, as we just explained, each comparison $\boldsymbol{y} \succ \boldsymbol{z}$ provides a directional hint in the solution space. Therefore, two preferences can also be redundant in the sense of suggesting similar directions.

- **Intra-case noise**: According to (5.1), preference feedback is correct only with a certain probability. Thus, even if unlikely, one may thoroughly observe $\boldsymbol{y} \succ \boldsymbol{z}$ although $\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^*) < \mathbf{P}(\boldsymbol{z} \succ \boldsymbol{y} \,|\, \boldsymbol{y}^*)$. According to what we just said, a preference of that kind will guide the search in the wrong direction and, therefore, could be considered as "noise".

- **Inter-case redundancy**: Instead of looking at a single preference, we now look at the whole set of preferences $\mathcal{P} = \mathcal{P}(\boldsymbol{x})$ that have been collected for a problem $\boldsymbol{x}$, because this is the information to be reused later on. Again, as explained above,

these preferences provide a "directional hint" in the solution space. Therefore, just like in the case of individual preferences, two sets of preferences $\mathcal{P}$ and $\mathcal{P}'$ can be redundant in the sense of suggesting similar directions in the solution space. Note that this type of redundancy is likely to occur for two problems having similar ideal solutions $\boldsymbol{y}^*$. Yet, even in that case the preferences are not necessarily redundant, because they might have been collected in different parts of the solution space.

- **Inter-case noise**: Just as a case may appear redundant in the context of other cases, it can be noisy in the sense that its preferences are inconsistent with those of the others. Here, inconsistency means that the preferences suggest very different directions in the solution space.

### 5.2.2 Integration of case base maintenance into Pref-CBR framework

In this section, we highlight where case base maintenance is integrated in our Pref-CBR framework, and show where it takes place during the problem-solving process.

- Pref-CBR search is performed as usual to solve a new problem $\boldsymbol{x}_0$ (lines 5–22 of Algorithm 3).
- The search yields a solution $\boldsymbol{y}^\bullet$ which is now stored and ready for usage in any of the proposed maintenance methods.
- In line 23 of Algorithm 3, the maintenance procedure is performed. The different types of maintenance strategies (inter-case redundancy, inter-case noise, intra-case redundancy and intra-case noise) test whether the case $\boldsymbol{x}_0$ is, or parts thereof are, considered to be redundant or noisy. Based on this result, the decision is taken of whether to store the solved case $\boldsymbol{x}_0$ along with its set of preferences collected during the search procedure preferences $\mathcal{P}_0$, or whether to delete the solved case if it is proven to be redundant or noisy. If the "intra-case" maintenance strategies are performed, it could be the case that only part of the case is deleted and not the whole case. This will be explained further in the following subsections.

**Algorithm 3** Pref-CBR Search($K$, $J$)

**Require:** $K$ = number of nearest neighbors collected in the case base
$J$ = number of preferences used to guide the search process

$\mathbb{X}_0 \leftarrow$ list of problems to be solved ▷ a subset of $\boldsymbol{X}$
$Q \leftarrow [\cdot]$ ▷ empty list of performance degrees
**CB** $\leftarrow \emptyset$ ▷ initialize empty case base
**while** $\mathbb{X}_0$ not empty **do**
  $\boldsymbol{x}_0 \leftarrow$ pop first element from $\mathbb{X}_0$ ▷ new problem to be solved
  $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\} \leftarrow$ nearest neighbors of $\boldsymbol{x}_0$ in **CB** (according to $\Delta_X$)
  $\{\mathcal{P}(\boldsymbol{x}_1), \ldots, \mathcal{P}(\boldsymbol{x}_K)\} \leftarrow$ preferences associated with nearest neighbors
  $\mathcal{P} \leftarrow \mathcal{P}(\boldsymbol{x}_1) \cup \mathcal{P}(\boldsymbol{x}_2) \cup \ldots \cup \mathcal{P}(\boldsymbol{x}_k)$ ▷ combine neighbor preferences
  $\boldsymbol{y}^\bullet \leftarrow \mathrm{CBI}(\mathcal{P}, \mathbb{Y})$ ▷ select an initial candidate solution
  $\mathbb{Y}^{vis} \leftarrow \{\boldsymbol{y}^\bullet\}$ ▷ candidates already visited
  $\mathcal{P}_0 \leftarrow \emptyset$ ▷ initialize new preferences
  **repeat**
    $\mathcal{P}^{nn} = \{\boldsymbol{y}^{(j)} \succ \boldsymbol{z}^{(j)}\}_{j=1}^{J} \leftarrow J$ preferences in $\mathcal{P} \cup \mathcal{P}_0$ closest to $\boldsymbol{y}^\bullet$
    $\mathbb{Y}^{nn} \leftarrow$ neighborhood $\mathcal{N}(\boldsymbol{y}^\bullet)$ of $\boldsymbol{y}^\bullet$ in $\mathbb{Y} \setminus \mathbb{Y}^{vis}$
    $\boldsymbol{y}^{query} \leftarrow \mathrm{CBI}(\mathcal{P}^{nn}, \mathbb{Y}^{nn})$ ▷ find next candidate
    $[\boldsymbol{y} \succ \boldsymbol{z}] \leftarrow \mathrm{Oracle}(\boldsymbol{x}_0, \boldsymbol{y}^{query}, \boldsymbol{y}^\bullet)$ ▷ check if new candidate is better
    $\mathcal{P}_0 \leftarrow \mathcal{P}_0 \cup \{\boldsymbol{y} \succ \boldsymbol{z}\}$ ▷ memorize preference
    $\boldsymbol{y}^\bullet \leftarrow \boldsymbol{y}$ ▷ adopt the current best solution
    $\mathbb{Y}^{vis} \leftarrow \mathbb{Y}^{vis} \cup \{\boldsymbol{y}^{query}\}$
  **until** convergence
  $q \leftarrow$ performance of solution $\boldsymbol{y}^\bullet$ for problem $\boldsymbol{x}_0$
  $Q \leftarrow [Q, q]$ ▷ store the performance
  Apply case base maintenance methodology ▷ inter-case redundancy/noise, intra-case redundancy/noise
  **if** ▷ case is redundant or noisy **then**
    **CB** $\leftarrow$ **CB** $\setminus \{(\boldsymbol{x}_0, \mathcal{P}_0)\}$ ▷ remove new experience (or a part of it) from the case base
  **else**
    **CB** $\leftarrow$ **CB** $\cup \{(\boldsymbol{x}_0, \mathcal{P}_0)\}$ ▷ memorize new experience
  **end if**
**end while**
return list $Q$ of performance degrees

## 5.3 Maintenance strategies

Our general maintenance strategy is incremental and essentially consists of deciding, for each new case $(\boldsymbol{x}_0, \mathcal{P}_0)$ produced, whether or not that case should be stored—and perhaps which parts thereof. To this end, each of the aforementioned types of noise and redundancy have to be handled in a proper way.

As already explained, the similarity or discrepancy between preferences or sets of preferences depends on the similarity or dissimilarity of the directional hints they provide. But how to quantify the latter? The direction suggested to the search process is a local property that depends on the current search state in $\mathbb{Y}$—as such, it is difficult to quantify in a single value. Reasoning on a more global level, the arguably most appropriate way to compare two sets of preferences $\mathcal{P}_1$ and $\mathcal{P}_2$ is to compare the respective globally optimal solutions $\boldsymbol{y}_1^{ML}$ and $\boldsymbol{y}_2^{ML}$, i.e, the likelihood estimates (5.2) with $\mathbb{Y}_0 = \mathbb{Y}$. Such a comparison could easily be done using $\Delta_Y$. However, finding the global likelihood maximizer might be very costly—this is why our search procedure is local. Besides, when comparing single preferences as a special case, the likelihood is often unbounded. In the following, we therefore propose approximate strategies that circumvent these difficulties and that are computationally more efficient.

### 5.3.1 Intra-case redundancy

Consider a case $(\boldsymbol{x}, \mathcal{P})$, and let $\boldsymbol{y}^\bullet$ denote the solution that the problem solving process ended up with—again, recall that $\boldsymbol{y}^\bullet$ will in general differ from $\boldsymbol{y}^*(\boldsymbol{x})$, either because the latter was not reached or because it may not even exist. Now, consider a single preference $\boldsymbol{y} \succ \boldsymbol{z}$ in $\mathcal{P}$. How redundant is that preference? To answer this question, we should compare the likelihood function (5.2) with and without the preference, i.e., the functions $\ell(\cdot \mid \mathcal{P})$ and $\ell(\cdot \mid \mathcal{P}')$ with $\mathcal{P}' = \mathcal{P} \setminus \{\boldsymbol{y} \succ \boldsymbol{z}\}$. Of course, comparing the functions globally is very difficult. Moreover, as explained above, we may not be able to compare their respective global maximizers either. What we could do, for example, is checking whether or not the locally restricted optimum in the neighborhood of $\boldsymbol{y}^\bullet$ would change, i.e., whether the local optimum for $\mathcal{P}$ is the same as the optimum for $\mathcal{P}'$. If not, then $\boldsymbol{y} \succ \boldsymbol{z}$ has an important influence and should certainly not be removed.

Figure 5.1 illustrates how the intra-case redundancy strategy is applied after a problem has been solved. To perform the check of removing case $\boldsymbol{x_0}$, we remove each

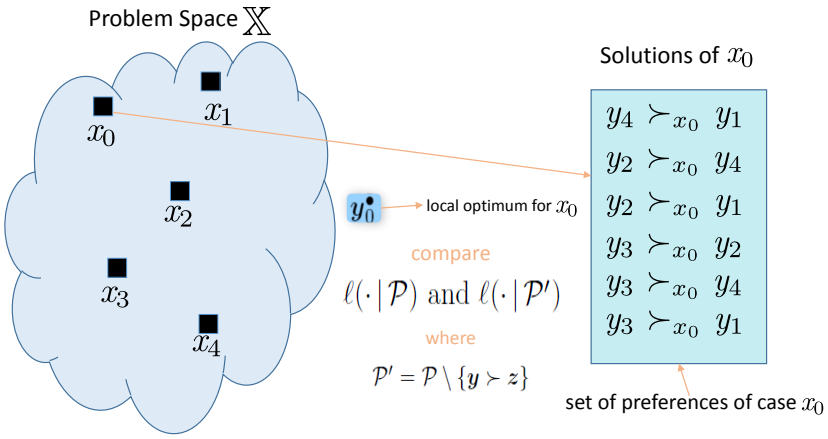


**Figure 5.1:** Intra-case redundancy strategy

pairwise preference in $\mathcal{P}$. If local optimum for $\mathcal{P}$ is the same as local optimum for $\mathcal{P}'$ then the pairwise preference could be considered as redundant and can be removed.

### 5.3.2 Intra-case noise

As explained earlier, we consider a preference $\boldsymbol{y} \succ \boldsymbol{z}$ as noise if $\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^*) < 1/2$. This property cannot be checked, however, because $\boldsymbol{y}^*$ is not known. Yet, using $\boldsymbol{y}^\bullet$ as a proxy, we could at least check if $\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^\bullet) < 1/2$. Figure 5.2 shows how to test which preferences are considered as noisy preferences, and accordingly delete them from the set of preferences $\mathcal{P}$ of the currently solved case. Check for each preference and see whether $\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^\bullet) < 1/2$, if this is the case then that pairwise preference could be considered as noise and can be removed.

### 5.3.3 Inter-case redundancy

Consider two cases $(\boldsymbol{x}_0, \mathcal{P}_0)$ and $(\boldsymbol{x}_1, \mathcal{P}_1)$ with solutions $\boldsymbol{y}_0^\bullet$ and $\boldsymbol{y}_1^\bullet$, respectively. How redundant are these cases or, more specifically, how redundant is the new case $(\boldsymbol{x}_0, \mathcal{P}_0)$ with respect to the previous case $(\boldsymbol{x}_1, \mathcal{P}_1)$? Again, for the reasons explained above, a comparison of the likelihood functions $\ell(\cdot \,|\, \mathcal{P}_0)$ and $\ell(\cdot \,|\, \mathcal{P}_1)$ or their maximizers may not be feasible. Instead, we again refer to the actually found solutions $\boldsymbol{y}_0^\bullet$ and $\boldsymbol{y}_1^\bullet$ as surrogates of these maximizers. More specifically, we compare the probability (5.2) of the preferences $\mathcal{P}_0$ under the associated (ML) parameters $(\boldsymbol{y}_0^\bullet, \beta)$ with the probability

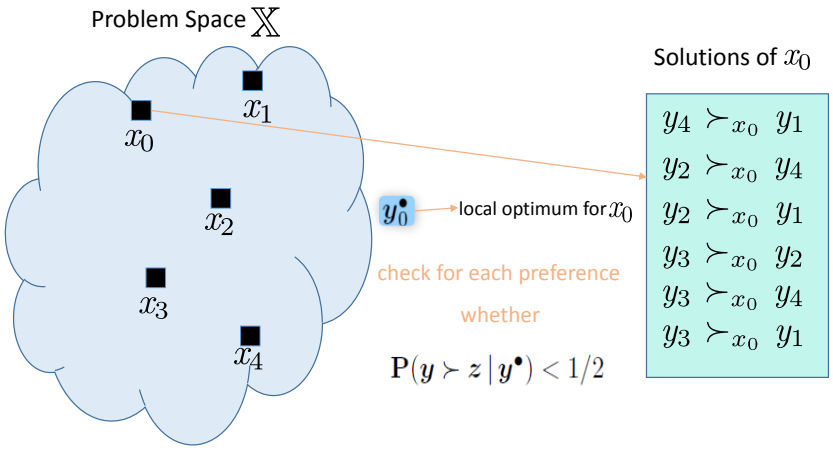


**Figure 5.2:** Intra-case noise strategy

under $(\boldsymbol{y}_1^\bullet, \beta)$, i.e., when replacing $\boldsymbol{y}_1^\bullet$ by $\boldsymbol{y}_0^\bullet$. If

$$\frac{\ell(\boldsymbol{y}_1^\bullet, \beta \,|\, \mathcal{P}_0)}{\ell(\boldsymbol{y}_0^\bullet, \beta \,|\, \mathcal{P}_0)} \geq t \tag{5.3}$$

for a threshold $t > 0$, this indicates that the preferences $\mathcal{P}_0$ are not only hinting at $\boldsymbol{y}_0^\bullet$ but also at $\boldsymbol{y}_1^\bullet$ (just like $\mathcal{P}_1$), which in turn can be interpreted as a sign of redundancy. Moreover, since not only the preferences but also the solutions themselves are reused, we additionally require

$$\Delta_Y(\boldsymbol{y}_0^\bullet, \boldsymbol{y}_1^\bullet) \leq v \tag{5.4}$$

for a second threshold $v \geq 0$. If both conditions are met, $(\boldsymbol{x}_0, \mathcal{P}_0)$ is considered redundant with respect to $(\boldsymbol{x}_1, \mathcal{P}_1)$. Our inter-case redundancy maintenance method can be described in detail as follows:

- Given a new problem $\boldsymbol{x}_0 \in \mathbb{X}$ to be solved, Pref-CBR is used to find a solution $\boldsymbol{y}_0^\bullet \in \mathbb{Y}$. In addition to the solution itself, Pref-CBR returns a set of preferences $\mathcal{P}_0$ (see Algorithm 3 for a detailed description of the problem solving process on the level of pseudo-code).

- To decide whether the new case should be stored, the $K$ nearest neighbors of $\boldsymbol{x}_0$ are retrieved from the current case base: $(\boldsymbol{x}_1, \mathcal{P}_1), \ldots, (\boldsymbol{x}_K, \mathcal{P}_K)$.

- The two criteria (5.3) and (5.4) are checked for $(\boldsymbol{x}_0, \mathcal{P}_0)$ and each of the cases $(\boldsymbol{x}_i, \mathcal{P}_i)$, $i = 1, \ldots, K$.

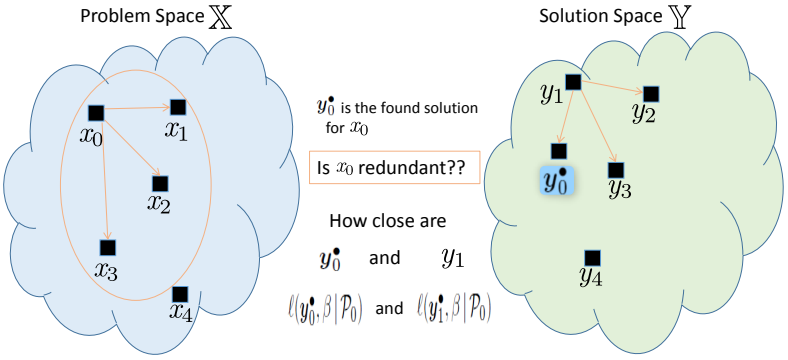


**Figure 5.3:** Inter-case redundancy strategy

- If the criteria are fulfilled for at least one of the $K$ cases, $(\boldsymbol{x}_0, \mathcal{P}_0)$ is considered as redundant and not stored; otherwise, it is added to the case base **CB**.

Note that this strategy has three parameters, namely, the number of neighbors $K$ and the thresholds $t$ and $v$ in (5.3–5.4).

The inter-case redundancy maintenance procedure is illustrated in Figure 5.3; if as previously mentioned, both conditions are met then the case $\boldsymbol{x}_0$ is a redundant case and can be safely removed from the case base without loss of useful information.

### 5.3.4 Inter-case noise

We just gave two conditions which, in conjunction, suggest the similarity (and hence the potential redundancy) of two cases. It is natural, then, to consider the cases as dissimilar if the opposite of at least one of the conditions holds, i.e., if either the ratio in (5.3) is smaller than some (small) threshold or the distance in (5.4) is larger than some threshold. If a new case $(\boldsymbol{x}_0, \mathcal{P}_0)$ is dissimilar in this sense to all of its neighbors, we may consider it as being exceptional or at least non-representative. If for example it is similar to all of its neighbors except for one neighbor, it might be that the older case already stored in the case base is a noisy case. Depending on the value of the ratios in 5.3 or in 5.4, either $(\boldsymbol{x}_0, \mathcal{P}_0)$ or $(\boldsymbol{x}_1, \mathcal{P}_1)$ is removed from the case base. To decide which case should be removed from the case base, we check which case is more dissimilar to its neighbors. If we find that our current case $\boldsymbol{x}_0$ is similar to all its neighbors except for the case $\boldsymbol{x}_1$ then we check the value of $\ell(\boldsymbol{y}_0^\bullet, \beta \,|\, \mathcal{P}_N)$ and $\ell(\boldsymbol{y}_1^\bullet, \beta \,|\, \mathcal{P}_N)$, where

$\mathcal{P}_N$ are the preferences of the neighbors of $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$. If $\ell(\boldsymbol{y}_0^\bullet, \beta \,|\, \mathcal{P}_N) \geq \ell(\boldsymbol{y}_1^\bullet, \beta \,|\, \mathcal{P}_N)$, then we remove the case $(\boldsymbol{x}_1, \mathcal{P}_1)$ from the case base since it is considered to be more dissimilar to its neighbors than $(\boldsymbol{x}_0, \mathcal{P}_0)$. If the opposite holds true, then we delete the case $(\boldsymbol{x}_0, \mathcal{P}_0)$.

A description of the inter-case noise maintenance method is as follows:

- Given a new problem $\boldsymbol{x}_0 \in \mathbb{X}$ to be solved, Pref-CBR is used to find a solution $\boldsymbol{y}_0^\bullet \in \mathbb{Y}$. In addition to the solution itself, Pref-CBR returns a set of preferences $\mathcal{P}_0$.

- To decide whether the new case should be stored, the $K$ nearest neighbors of $\boldsymbol{x}_0$ are retrieved from the current case base: $(\boldsymbol{x}_1, \mathcal{P}_1), \ldots, (\boldsymbol{x}_K, \mathcal{P}_K)$.

- The two criteria (5.3) and (5.4) are checked, but with the signs before the thresholds being reversed as well as the thresholds having different values, for $(\boldsymbol{x}_0, \mathcal{P}_0)$ and each of the cases $(\boldsymbol{x}_i, \mathcal{P}_i)$, $i = 1, \ldots, K$.

- If the criteria are fulfilled for all of the $K$ cases, $(\boldsymbol{x}_0, \mathcal{P}_0)$ is considered as noisy and should not be stored; otherwise, it is added to the case base **CB**. If the criteria fail to be fulfilled for all except one of the nearest neighbors, to make sure which case is more noisy, the current case $\boldsymbol{x}_0$ or the older stored case from the $K$ cases, we can perform a simple check. $\ell(\boldsymbol{y}_0^\bullet, \beta \,|\, \mathcal{P}_N) \geq \ell(\boldsymbol{y}_1^\bullet, \beta \,|\, \mathcal{P}_N)$ for example indicates that the case $\boldsymbol{x}_1$ is more dissimilar to the $K$ cases and is thus removed.

The preceding description of the inter-case noise maintenance strategy is shown in Figure 5.4. If the conditions are satisfied for all nearest neighbors, it means that a case is noisy, in other words it is different from its nearest neighbors. If the conditions are satisfied but fail for one case of the nearest neighbors, it means that either the noisy case is the currently solved one, or actually the noisy can be its nearest neighbor, which is already stored in the case base. For that reason, the likelihood of each of the two cases, $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ is compared, given all preferences of their nearest neighbors, as explained above. The case with the lowest likelihood given the preferences of the nearest neighbors, is considered to be the noisy case and can be removed.

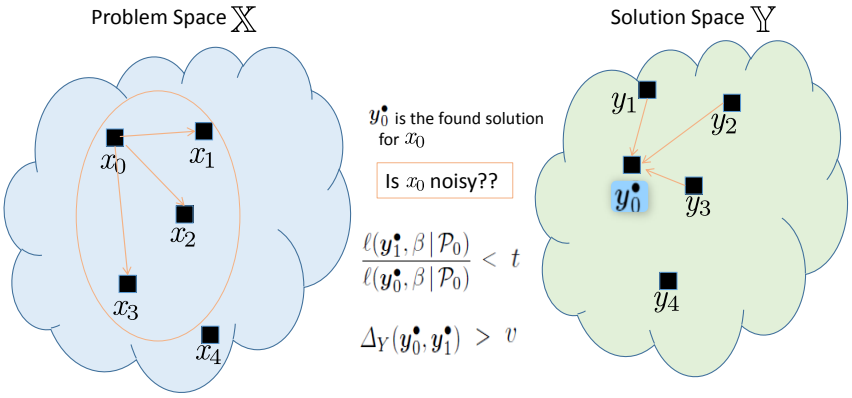


**Figure 5.4:** Inter-case noise strategy

## 5.4 Conclusion

This chapter extends our framework of preference-based CBR by methods for dynamic case base maintenance. The main goal of these methods is to increase efficiency of case-based problem solving while maintaining performance. The effectiveness of our four maintenance approaches will be illustrated in a case study with the traveling salesman problem, shown in detail in Chapter 7.

The implementation of our maintenance method consists of the strategies discussed in Section 5.3. These include a strategy for handling what we mentioned as inter-case redundancy, inter-case noise, intra-case redundancy as well as intra-case noise. The first two strategies handle whole cases, while the latter two strategies handle parts of cases (pairwise preferences within a case).

# 6

# Related Methodologies

In the previous chapters, we introduced and discussed in detail our Pref-CBR framework and also showed in each successive chapter how components were added to the framework for increasing its performance and efficiency. Learned similarity measures described in Chapter 4, led to increased performance of the search, and the different case base maintenance strategies listed in Chapter 5, increased the efficiency of the CBR system while maintaining its performance.

In this chapter we discuss and show several methodologies which are in some way related to our Pref-CBR problem-solving framework. In the following sections we describe each methodology, explain it and relate it to our work. The following methodologies consist of different search methods (stochastic population-based search, black box search and heuristic search), as well as machine learning approaches (output space search in machine learning, machine learning with human in the loop and reactive search using intelligent optimization). We describe how these different methodologies and approaches relate to our framework, and we highlight the similarities and differences between these approaches and our Pref-CBR problem-solving framework.

Having learned how the Pref-CBR problem-solving framework operates, as well as looking more deeply into its embedded components from the previous chapters, we can now imagine what our framework mainly consists of. The basic components of the framework are: case-based reasoning, preference-based knowledge representation, the search and machine learning methods. In Chapter 2 we took a close look at CBR and in Chapter 3, we described the preference-based knowledge representation and we also discussed the related previous work. In this chapter we will discuss the different search

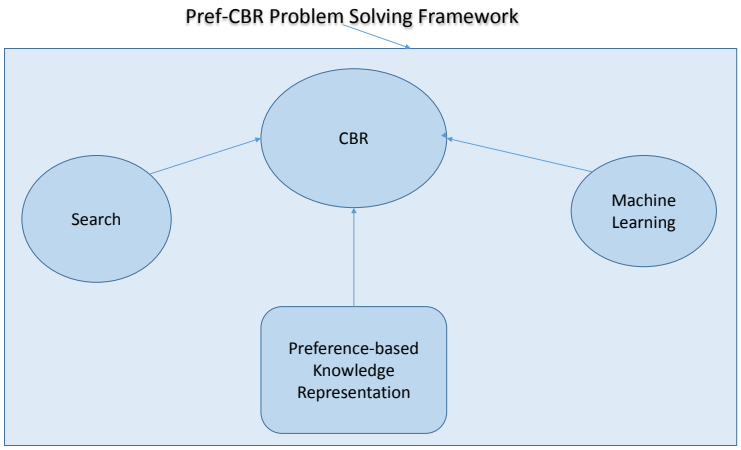


**Figure 6.1:** Pref-CBR framework and related methodologies

strategies which relate to our framework, and we will also look at how machine learning approaches work and how these also relate to our framework as well. In Figure 6.1, a simple illustration shows the basic components of the Pref-CBR framework.

## 6.1 Heuristic search

Heuristic search is a method designed for solving problems which cannot be easily solved to get an exact solution. As we use heuristic search in our Pref-CBR framework, it is plausible to look at some heuristic search methods. Many applications in engineering and industrial design, many business activities, and even frequent activities such as internet routing or holiday planning, all require optimization in some way [108]. The aims of optimization can differ between minimizing energy consumption and costs, or maximizing profit, output, performance or efficiency. As the author in [108] states, as real world applications have limited resources, money and time, it is essential to find solutions to optimally use these valuable resources under constraints, by using various suitable efficient search algorithms.

Heuristic search can thus be used to find an approximate solution to such complex problems, when classic search methods would take a very long time. Consequently, this is achieved by trading optimality, completeness, accuracy, or precision for speed. In a way, it can be considered a shortcut; heuristic search can be used when finding an optimal solution is very complex and can take a very long time. The objective of a

heuristic is then to produce a close-to-optimal solution in a short time, which would be satisfactory for the problem at hand.

Although heuristic methods do not guarantee an optimal solution for a given problem, they can produce an acceptable solution within a reasonable amount of time [109]. Heuristic-based algorithms include some well known search methods such as: hill climbing, greedy search, A* algorithm, tabu search, simulated annealing, genetic algorithms and particle swarm optimization. In artificial intelligence, as well as in operations research and many other fields, the need for solving difficult combinatorial problems has led to the use of heuristic search methods [110].

The complex optimization problems, including a vast number of possibilities for finding a solution, occur in many fields such as knowledge-based systems, design robotics, scheduling and pattern recognition. As stated in [110], for the formerly mentioned problems, an attempt to generate all relevant alternatives by computer would be hopeless. Since we use heuristic search in our framework, it is reasonable to discuss some heuristic search methods and relate them to our Pref-CBR framework.

### 6.1.1 Best-first search algorithms

Best-first search is a general heuristic search algorithm, which always expands next a node of lowest cost [111]. The choice of which node to expand next is provided by an *evaluation function* that returns a value describing the desirability of expanding the node. The nodes are ordered so that the one with the best evaluation (according to an evaluation function) is expanded first [112]. As the authors in [112] state, the former search strategy aims at finding low-cost solutions and in order to focus the search, the aforementioned measure must include some estimate of the cost (distance or time) of the path (a sequence of edges which connect a sequence of vertices) from a state (a unique configuration of information) to the closest goal state (solution). Let us now look at the following search strategies, which incorporate *best-first search*.

#### 6.1.1.1 Greedy search

A best-first search that minimizes a function $h$, estimating the cost of the cheapest path from the state at node $n$ to a goal state, to select the next node to expand, is called greedy search [112]. Greedy best-first search expands the node which appears to be *closest* to the goal. Figure 6.2 illustrates how greedy search works, by having several

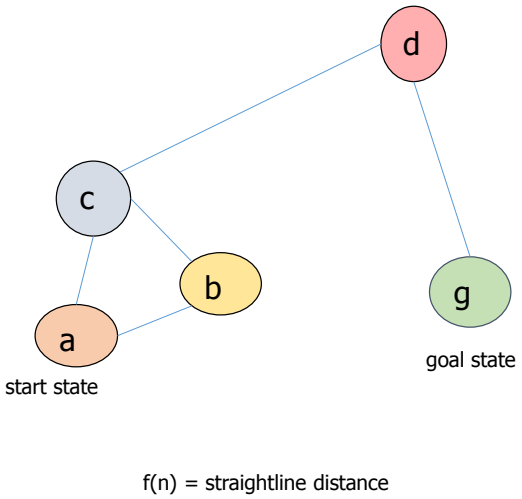


**Figure 6.2:** Greedy best-first search

nodes, the algorithm chooses always the shortest path to the next node. Each time a node is chosen, the algorithm chooses the next node to reach by choosing the shortest path to the next node. The assumption is that by choosing the shortest path each time, the immediate best choice is made without knowledge of which path from the beginning is the shortest, and usually this is a good strategy. Greedy algorithms tend to find solutions quickly; the solutions might not be optimal (this would take a more careful analysis of long-term options) but they often perform well [112]. The search can be considered incomplete because it can get stuck in loops, as shown in Figure 6.2.

#### 6.1.1.2 A* algorithm

Greedy search aims at minimizing the estimated cost to a certain goal, but does not consider minimizing the cost of the path taken so far to reach that goal during the search. A* search combines both goals, minimizing estimated cost to the goal h($n$) and minimizing the cost of the path from the start node to node $n$ (g($n$)). It can then be said that A* algorithm minimizes the total path cost f($n$) = h($n$) + g($n$) to efficiently compute optimal solutions [113]. If h($n$) is consistent, A* search is considered to be complete and optimally efficient on locally finite graphs. Unfortunately, for most problems the number of nodes within the search space is exponential in the length of

the solution, thus A* can run out of space since it keeps all generated nodes in memory [112]. Since at least the entire open list during a problem-solving episode must be saved, A* algorithm is severely space-limited in practice as the best-first search algorithms are in general [113].

As we have seen, best-first search algorithms can find efficiently optimal solutions quickly, in practice they have the problem of using a huge amount of memory due to the many trials that are being performed before reaching a solution. In our Pref-CBR framework this problem does not take place since we assume that our trials are costly and there is only a limited number of trials during our problem-solving episodes.

### 6.1.2 Iterative improvement algorithms

Iterative improvement algorithms often provide the most practical approach in solving problems which have the property that, the state description contains all the information needed for a solution. The general idea is to start with a complete configuration, set of values of parameters, and by using iterative improvement algorithms, modifications are made to improve the quality of the configuration [112]. As [112], the search starts with a random configuration and repeatedly considers various moves where some are accepted and some rejected depending on the evaluation function of the state at that point, trying to find optimal solutions. The algorithm basically constructs all neighbors (or a given number of neighbors in case of iterated local search) and selects the best one; one moves from one solution to a better one in an intelligent way [114].

#### 6.1.2.1 Hill-climbing search

The hill-climbing search algorithm is simply a loop that continuously moves in the direction of the increasing value, when there is more than one best successor to choose from, the algorithm selects among them at random [112]. Hill-climbing search is a form of local search, which first starts with a random guess of a solution and tries to optimize an objective function by selecting any local change that improves the current value of that objective function [115]. The search terminates once no local move could further improve the objective function and upon termination the search would have reached a local but not necessarily a global optimum for the objective function [115]. Hill-climbing search has a long history in the area of continuous optimization; in the continuous search spaces, the gradient of the objective function (gradient descent) is

used to take local steps in the direction of the greatest possible improvement [112, 115]. As stated by [112], a well-known drawback of the former search method is that it can get stuck at a local maxima and the algorithm will halt even though the solution may be far from satisfactory. This problem can be avoided by the next method which we will describe, the tabu search.

#### 6.1.2.2 Tabu search

Tabu search exploits data structures of the search history to decide on the next moves, by looking at a *tabu* list which stores attributes of the previous few moves [109]. The tabu list is of limited length and is updated after each local move [115]. The major distinction of tabu search to the other search methods is that it uses a short-term memory (set for release of the tabu status which restricts certain areas from being searched) and a long-term memory (which stores frequency of searching in each area). Instead of terminating when reaching a point of local optimality, tabu search operates its embedded heuristic to continue by forbidding moves with certain attributes *making them tabu* and choosing moves which heuristic assigns to a highest evaluation [109, 110]. As explained by [116], tabu search is an extension of classical local search methods, and the first two basic elements of tabu search heuristic are the definition of its search space and its neighborhood structure. The most commonly used *tabus* involve recording the last few transformations performed on the current solution and prohibiting reverse transformations; this is to avoid getting stuck at a local optima [116]. As stated by [115], a tabu list provides a powerful way of forcing a local search method to explore a larger part of the search space.

As we can see from the description of how the iterative improvement algorithms operate, we can see how our Pref-CBR framework search process is similar in the way we improve an initial solution that we start out with, for a given problem. We start out with an initial solution and we keep improving it until we end up with an optimal solution, it can of course be an optimal local solution and not a global one, as the formerly described methods. There are some major differences though between our search procedure and the former methods. One major distinction of our search process is that the initial solution we start out with is not randomly chosen, our initial solution which we further improve is actually obtained based on the stored preferences of the previously solved nearest neighbor cases and not a random choice. The second major distinction, which in

fact is a significant advantage, is that we take advantage of repetitive problem solving and we use stored knowledge in our case base (previous solutions) to get our initial solution which we improve and needless to say this initial solution improves over time as we have more cases stored in the case base. This leads to overall improvement of the performance. All previously mentioned approaches use only the last episode information and not previously stored experience. We store experience and start the search, unlike all other approaches mentioned, which basically start searching from scratch. Another difference is that we are guided in our search by the knowledge gained from the pairwise comparisons rather than using an evaluation function; we move on in the search by qualitative feedback information from the pairwise comparisons rather than absolute values of an evaluation function h($n$) assessing the cost at each state $n$. Evaluation functions might be noisy, when each function evaluation is subject to random noise due to certain experimental settings including simulation or approximate solutions of a numerical problem [117]. An advantage of our Pref-CBR framework is that although we might have some noisy preferences, this problem is overcome by the maximum likelihood approach we use in our inference step for choosing a solution.

### 6.1.3 Nature-inspired optimization algorithms

The previous search methods described above eventually seek to solve optimization problems, but population-based algorithms help the search not to get stuck at a local optima, making them attractive to use for providing good solutions to a wide range of complex optimization problems. Natural optimization often involves variation methods (genetic mutations or slight parameter shifts), parallel optimization scenarios (agents exploiting local solutions), adaptive strategies (if environment changes over time, optimization adapts), partial exploitation of the solution space and providing different but equally optimal solutions. It is important to note that in our Pref-CBR framework, optimization plays a major role since we continuously try to optimize our solution based on the preferences given by the oracle. We will describe the three most well known nature-inspired optimization approaches in this section (genetic algorithms, simulated annealing and particle swarm optimization). We will look at the similarities and differences between the latter approaches and our Pref-CBR framework.

#### 6.1.3.1 Genetic algorithms

Simply stated, genetic algorithms are probabilistic search procedures designed to work on large spaces which involve states that can be represented by strings [118]. These algorithms are propagating the best new offspring from the parent population, thereby proceeding in an evolutionary fashion which encourages the survival of the fittest [110]. Starting from an initial population, a set of parents are chosen according to a fitness function, to breed a new generation of candidate solutions. According to [109], each parent contains a set of *chromosomes* which are the desirable features, and the offspring are reproduced from two parents by mixing parts of chromosomes of each parent in a *crossover* fashion. This transmits good features of parents into the next generation. Accordingly, the efficiency of a genetic algorithm depends highly on the choice of the fitness function, the representation of the desirable characteristics in the chromosomes and the appropriate use of the crossover mechanism and mutation (for preserving and introducing diversity to avoid local minima).

#### 6.1.3.2 Simulated annealing

As the name of the algorithm indicates, it is derived from the intent to pattern its approach after the physical process of annealing [110]. Annealing is the process of reducing the temperature of a material to its minimum state of energy, which is called thermal equilibrium. Described by [119], simulated annealing explores a function's entire surface and tries to optimize the function while moving through the space in both uphill and downhill directions. It is a hill-climbing algorithm, which can accept during the search an inferior solution in the neighborhood, to escape local maxima, according to a probability function [109]. Simulated annealing tries to inject just the right amount of randomness to escape local maxima early in the search process without getting off course later in the search, when a solution is nearby. At the beginning of the search, many widely distributed positions of the given function are probed around the currently found maximum value and this allows for sometimes choosing an inferior solution to escape a local maxima, based on an *acceptance probability*. Following the idea of the annealing process where the temperature of annealing is gradually reduced, the probability in the aforementioned function is set to be high at the beginning of the optimization process and is gradually reduced to zero. Following a function called the

*cooling schedule*, the rate of the drop of the acceptance probability is controlled and as the temperature cools to a predefined threshold, a solution is reached. A theoretic property of simulated annealing is that, if the temperature is annealed sufficiently slowly, there can be a guarantee to find an optimal solution. The efficiency of the algorithm as well as the quality of the solution depend on the choice of the *cooling schedule*.

#### 6.1.3.3 Particle swarm optimization

The particle swarm is also a population-based stochastic algorithm used for optimization, where all population members survive from the beginning of a trial until the end. The particle swarm does not use selection but rather the interactions of all members result in iterative improvement of the quality of problem solutions over time [120]. The initial ideas on particle swarms of Kennedy (a social psychologist) and Eberhart (an electrical engineer) exploited analogues of social interaction, they involved analogues of bird flocks searching for corn; these turned later into a powerful optimization method, which is called particle swarm optimization [121]. As the authors in [121] explain: "the *particles* are placed in the search space of some problem or function, and each evaluates the objective function at its current location. Each particle determines its next move based on its current and best-fitness locations with those of one or more members of the swarm, until eventually the swarm as a whole like a flock of birds collectively foraging for food, is likely to move close to an optimum of the fitness function". It requires only primitive mathematical operators, and is computationally inexpensive in terms of both speed and memory requirements [120]. Particle swarm optimization uses the concept of *fitness*, as do all evolutionary computation paradigms, and its adjustment for the best fit is similar to the *crossover* operation utilized by genetic algorithms [120].

The important thing to notice about all these nature-inspired optimization algorithms, is that the goal is to optimize a solution for a given problem, based on a "fitness function". In our Pref-CBR framework we also optimize a solution for a given problem, and we move in the solution space relying on qualitative feedback based on the generated preferences. A property we share with the above mentioned nature-inspired algorithms is that we cannot guarantee to find the global optimum. An advantage of our framework is the existence of the pairwise assessment by an expert, which does not

exist in nature-inspired algorithms. We also have the advantage of simultaneous problem solving episodes (gaining knowledge from previous cases), as opposed to parallel optimization of above methods. An advantage of the above methods though is that the solution modification operator adapts over time, as in simulated annealing, while in our framework this does not take place (most parameters are fixed) except with the triggering of the learning of similarity measures module.

### 6.1.4 Black box search

Black box search basically as its name implies, is based on some automatic approaches which can optimize the performance of a given learning algorithm to operate or solve a task required at hand [122]. The key property of black box optimization is that a good optimal solution is predicted, but with no explanation of *why* this output was given. The problem-solving agent has only information about the goal test and the heuristic function which are used as *black boxes* to achieve the goal; the agent cannot *look inside* to select actions which would be useful in achieving that goal [112]. This search method is worth mentioning, as it resembles our Pref-CBR search process, and it is called Bayesian optimization. Instead of using standard search methods, Bayesian optimization can efficiently trade off exploration and exploitation of the parameter space and quickly guides the user to the configuration that best optimizes some overall evaluation criterion like accuracy or likelihood by automatic tuning of the parameters [122]. Bayesian optimization techniques have been successfully applied to planning, robotics, sensor placement, advertising, recommendation, intelligent user interfaces and automatic algorithm configuration [123]. Such tuning of parameters can be considered as the optimization of an unknown black-box function for expensive function evaluations, as they involve running the primary machine learning algorithm to completion. As the authors suggest in [122]: "in a setting where function evaluations are expensive, it is desirable to spend computational time making better choices about where to seek the best parameters. Bayesian optimization provides an elegant approach and has been shown to outperform other state of the art global optimization algorithms on a number of challenging optimization benchmark functions". Automatic black-box optimization methods greatly reduce time-consuming design processes which require human intervention and providing of human expertise, therefore Bayesian optimization is particularly

suited for robotic applications, where it is crucial to find a good set of parameters in a small number of experiments [124].

As we have learned about this Bayesian optimization (black box search), we can now mention the difference between it and our Pref-CBR search. In black box search, the tuning of the parameters is done automatically, we do not know what the best parameters are but they are automatically adjusted to create a desired target function. A typical example of representing a black box search are neural networks, there are many parameters involved to create a target function which are adapted but without interpretation of the emerging configuration of these networks. In our Pref-CBR framework we have known parameters (most of them being fixed) such as a radius of a neighborhood, the number of neighbors, locality of the neighborhood, etc. The learning in our framework does not take place in the algorithm parameters, but rather in the stored examples. During the search, our oracle might be giving the preferences based on the importance of some parameters more than others, but we do not know this information and the search is adjusted according to the oracle's preferences by optimizing the likelihood of a solution given those preferences. That is why we use the Bayesian method in particular for our learning of similarity measures for the solutions, described in detail in Chapter 4. The second commonality between our approach and the Bayesian optimization, is the assumption that expert knowledge is expensive to obtain, thus there is a limited number of queries to the oracle. In both approaches, the former and the latter, there is the advantage of using past experience in finding a solution to a new problem. One difference between the two approaches though is that in the Bayesian optimization the automatic parameter tuning is done based on quantitative feedback, while in our approach we only have qualitative feedback from the preferences.

As we use heuristic search in our Pref-CBR framework, we believed it is important to discuss some heuristic search methods and compare them to our search strategy. Forming the neighborhood around the solution for adaptation of the solution and improving it, is the local search embedded in our framework. The maximum likelihood is then used to find the best solution from the neighborhood, given the preferences of the nearest neighboring problems. Having improved the solution, a neighborhood is formed again and the same search procedure is repeated. Further details of the problem solving process are explained in Chapter 3. We have listed in the previous section some search methods and we highlighted the similarities and differences between them and

our Pref-CBR search. We have also shown the advantages of our Pref-CBR framework over other commonly used search methods. In the next section we take a look at the concept of machine learning and how it is used for learning, since learning is also a part of our Pref-CBR framework. We also use machine learning in our inference procedure (maximum likelihood estimation) as well as in our learning of similarity measures' component (Bayesian inference and the perceptron algorithm).

## 6.2 Machine learning

The machine learning algorithms are able to perform important tasks by generalizing and learning from examples; they are widely used in computer science and other fields [125]. As explained by [125]: "machine learning systems automatically learn programs from data, making it very useful to apply in web search, span filters, recommender systems, ad placement, credit scoring, fraud detection, stock trading, drug design and many other applications". In this section, we will mention generally what machine learning is used for as well as some approaches in machine learning which in some way resemble our Pref-CBR framework.

There are several learning methods used in machine learning which are: supervised learning (input data is called training data and has a known label or result such as spam/not-spam), unsupervised learning (input data is not labeled and does not have a known result), and semi-supervised learning (input data is a mixture of labeled and unlabeled examples). Every learner must embody some knowledge or assumptions beyond the data it is given, in order to generalize beyond it. As described by authors in [125]: "machine learning is not magic; it cannot get something from nothing, but it can get more from less". The most mature and widely used machine learning type is classification, where a classifier inputs a vector of discrete or continuous feature values and outputs a single discrete value, a *class*. Another type of learning is the structured prediction, which is discussed further in the next subsection.

### 6.2.1 Machine learning output space search

Structured prediction or structured (output) learning is a proper subclass of supervised machine learning. This structured output learning includes techniques which involve predicting structured objects, instead of prediction of scalar discrete or real values.

The task of inferring a function from labeled training data in machine learning is called supervised learning, where the training data consists of a set of training examples. In this subsection we will mention specifically how output space search for structured prediction is performed, as well as show the relation between our Pref-CBR framework and the structured prediction via output space search.

One of the key challenges in machine learning, is learning the general functional dependencies between arbitrary input and output spaces [126]. Structured data is data which consists of several parts containing information that relates these parts of data together (text, audio, folds of a protein or images). Structured output prediction is predicting a structured output from input data, in contrast to predicting just a number from classification or regression. To search the space of complete structured outputs, the following procedure takes place: given an input and guided by a learned cost function, the least cost output is thus uncovered during the search (e.g., best-first or greedy search), and returned at the end of the search [127]. Some examples of tasks where the inputs and outputs are structured objects are: information extraction, scene understanding, part-of-speech tagging and image scene labeling [128]. As stated by [127], in most search-based approaches to structured prediction, the true loss function of the structured prediction problem is used to guide the search. Some applications where structured output prediction is applied are natural language processing (output: sentences), bioinformatics (output: bipartite graphs), speech processing (output: audio signal), and robotics (output: sequence of actions).

The following works describe some approaches to structured output prediction using different methods. In the work of [126], they address the issue of designing classification algorithms which can deal with complex outputs, such as trees, sequences, graphs or sets. They consider problems which involve multiple dependent output variables, structured output spaces and classification problems containing class attributes. Contrary to the idea of learning a cost function to score the structured outputs by [127], the authors in [126] address dealing with more complex output spaces by extracting combined features over inputs and outputs. For a large class of structured models, the work of the latter proposes a support vector machine algorithm which allows the learning (in polynomial time) of mappings that involve complex structures. The authors in [126], show that a key advantage of their algorithm, is the flexibility to include different loss functions which optimize directly the desired performance criterion. Similar also

to the latter work is the method for structured output tracking by [129]; they use a kernelized structured output support vector machine that is learned online to provide adaptive tracking. This tracking method is widely used in computer vision for tracking arbitrary objects, some wide-ranging applications include human-computer interaction, surveillance, augmented reality, scene understanding and action recognition.

Recalling our Pref-CBR framework, where the output (solutions) are predicted from the input (problems), and using a cost function for scoring the solutions (our oracle which gives the preferences and scoring of the solutions given by our inference method), the relation to the machine learning output space search can be seen. In both cases also, it is a time-bounded search, where the least cost output is returned at the end of this search process. In both cases, the solution space may be structured, but in our Pref-CBR framework we have the advantage of dealing with such structured outputs by just defining a good similarity measure and we also have the advantage of previously stored knowledge that can be used for new cases to be solved. Structured prediction has the advantage though of generating de novo solutions, for example generating a new drug design, where in CBR this design may be difficult to generate when there is a limited set of previously stored examples.

### 6.2.2 Machine learning with human in the loop

One of the biggest issues with machine learning is that it is often very easy to get an algorithm to 80 percent accuracy but nearly impossible to get an algorithm to 99 percent accuracy. The "human in the loop" computing solves this issue by using the human judgment, to be fed back into the algorithm to make it smarter. In this section, we will describe how the "human in the loop" affects the performance of a machine learning algorithm, as well as mention some applications where this aforementioned computing method has been used. We will also relate this computing method to our Pref-CBR framework.

#### 6.2.2.1 Human in the loop

There are several ways of including a *human in the loop* in machine learning algorithms. One way is to use a human as a means for information extraction, where users are allowed to specify the nature of the information structures they desire. As explained by [130], the strengths of humans and machines can be combined in the following way:

the human is proficient at judging an information structure as desirable or undesirable, while the machine is proficient at quickly and efficiently locating similar examples from large quantities of data. Another way of using a *human in the loop* is to apply programming by feedback, which involves a sequence of interactions between the active computer and the user, in which the user provides preference judgments on pairs of solutions supplied by the active computer [131]. Another approach includes having a human user going one step further and not only provide feedback about past actions, but also provide future directed rewards to guide subsequent actions [132]. Users can be included in an interactive machine learning model for classification, which allows users to train, classify/view and correct the classifications [133, 134].

Another view of including a *human in the loop* is suggested by [135]. They propose to change the limitations of present day technology, by engaging machines implicitly and indirectly in a world of humans; computers would be put in the human interaction loop, rather than the other way around. Multiple audio-video sensors can be attached in what is called *Computers in the Human Interaction Loop* rooms, to "observe" humans and can then be analyzed. As explained by [135], the analysis of all audio-video signals in the environment (speech, faces, signs, bodies, gestures, objects, attitudes, events and situations) provide answers, which allow computers to engage and interact with humans in a human-like manner.

#### 6.2.2.2 Applications of human in the loop in machine learning

Many safety-critical systems are interactive, they interact with a human being, and the human operator's role is central to the correct working of the system [136]. Examples of such interactive systems (human-in-the-loop control systems) include fly-by-wire aircraft control systems (interacting with a pilot), automobiles with driver assistance systems (interacting with a driver), and medical devices (interacting with a doctor, nurse, or patient). Another type of interactive systems is presented by [132], which integrates machine learning and human-robot interaction. In the latter system, the reinforcement learning algorithm benefits from the human-robot interaction and learns how humans teach the robots. Accordingly the learning algorithm modifies the action selection mechanism, and there is a significant improvement in the learning performance of the agent when the robot is tested later in a second study. Learning from humans is also applied in assistance systems (e.g. email categorizing, conference planning),

where humans are in the loop for both the learning and evaluation steps [137]. These assistance systems consist of multiple machine learning components, natural language processing and optimization techniques.

A *human in the loop* approach has been also used for image characterization for medical images. In this image characterization approach, an expert radiologist in each anatomic region, selects images for the database and provides differential diagnosis and includes treatment information. This information can be useful to a less experienced practitioner, enabling him/her to use the stored expertise, and provide the role of an expert consultant if confronted with a similar image [138]. [134] apply the *human in the loop* in visual recognition of images, by asking users some questions and accordingly classify correctly the object in the image. In the work of the latter authors, they show that this interactive, hybrid human-computer method for object classification, drives up recognition accuracy to levels that are not only acceptable, but can be considered good enough for practical applications. Machine learning in interactive settings is also proposed by [133], where machine learning and computer vision techniques are applied for image classification. The goal is to have a human user upon receiving an image, to do some manual classification, thus training a classifier. The user can then later refine the classifier by adding more manual classification, if the classifier is still not satisfactory. The proposed method by [133] replaces the analysis of many feature combinations by the machine learning algorithm, especially if there are many features, by an interactive machine-learning model that allows users to train, classify/view and correct the classifications.

It is clear that having a human in the loop can aid the learning in machine learning algorithms. The benefits of having a *human in the loop* has been stated; whether the feedback is given by the user in an interactive manner, continuously updating the learning process, or whether the feedback of the user is during testing of the algorithm for improving the learning. In our Pref-CBR framework, our *oracle*, is the *human in the loop* component of our algorithm. Although this oracle does not necessarily represent a human, it represents an expert of some form. It can alternatively be a human or some expert program, which in turn also represents expertise knowledge. We can conclude that the human in the loop can be a special case in our Pref-CBR framework, where our framework is more generic and can operate also without necessarily having a human in the loop. Our image correction application, described in detail in the proceeding

chapter shows a nice example of a human in the loop integrated in our Pref-CBR search process.

### 6.2.3 Reactive search and intelligent optimization

A major part of machine learning is the *learning*, thus it is plausible to state the question of: what is learning and how does it come about? "Learning takes place when the problem at hand is not well known at the beginning, and its structure becomes more and more clear when more experience with the problem is available. Human problem solving is strongly connected to learning, and in addition to learning, search by trial-and-error, generation, tests and repeated modifications of solutions by small local changes are also part of human life. What is critical for humans is also critical for many human-developed problem solving strategies. It is not surprising that many methods for solving problems in artificial intelligence, operations research, and related areas follow the search strategy of adding one solution at a time from a tree of possibilities, or by searching from a formed trajectory of candidate solutions on a landscape defined by the corresponding solution value" [139]. In reactive search the history of the search and the knowledge accumulated while moving in the configuration space is used for self-adaptation in an autonomic manner; the algorithm maintains the internal flexibility required to address different situations during the search, but what is automated is actually the adaptation, and it is executed while the algorithm runs to solve a single instance while reflecting on its past experience [140].

#### 6.2.3.1 Parameter tuning in heuristics

Most local search-based heuristics, such as tabu search and simulated annealing, although very efficient and useful in many practical applications, they are extremely sensitive to their own internal parameters [141]. The optimal parameter value can differ according to the problem instance being solved and the data used, therefore, the same algorithm might require some precise fine tuning in order to be applied to a new problem. As clearly explained by [139], parameter tuning is a crucial issue both in the scientific development and in the use of heuristics in practical applications. In some cases the detailed tuning is executed by a researcher or by a final user. As a result, the reproducibility of the heuristics results is difficult, as is comparing different parameters

of algorithms. As the authors in [139] suggest, there are some machine learning methods which can be profitably used in order to automate the tuning process and make it an essential and fully documented part of the algorithm. Reactive search optimization leads to dynamic adjustment of search parameters, thus leading to faster overall optimization time as well as an enhanced reproducibility of performance results [142]. Before an algorithm is presented to the scientific community, the algorithm designer faces a long and hard development phase, a possible exploratory phase of preliminary tests followed by an exhaustive documentation of the tuning process [139]. As the name suggests, *reactive* hints at a ready response (a reaction), to events during the search through an internal *feedback loop for online self-tuning and dynamic adaptation* [139].

#### 6.2.3.2 Reacting on the neighborhood

A basic problem-solving strategy consists of first starting with an initial solution (randomly or intelligently chosen), and then repeatedly improving this initial solution by small steps, refer to Subsection (6.1.2). At each repetition the current configuration is slightly modified, the function to be optimized is tested, the change is kept if the new solution is better, otherwise another change is tried and tested. As stated by [139], the idea is that if one starts at a good solution, solutions of similar quality can, on the average, be found more in its neighborhood than by sampling a completely unrelated random point. To avoid the search to fall into local optima, diversification methods are required such as including a tabu list in tabu search, having a random acceptance criteria in simulated annealing, perturbation operator in iterated local search or having multiple neighborhoods in variable neighborhood search [143]. Some problems exist, for which no known algorithms are available which can ensure optimality in a reasonable amount of time, this led to the motivation of the development of alternative methods to obtain an acceptable solution from a practical point of view (a good solution in a reasonable amount of time); one of these methods is the local search [144]. It is then important as suggested by [139] to decide on a set of local moves to be applied (neighborhood) as well as deciding on a way to pick one of the neighbors to be the next point during the search procedure.

Looking back at the reactive search and intelligent optimization, the relation to our Pref-CBR framework can be clearly shown. In our Pref-CBR framework, the history of the search, which is the knowledge stored in the case base, is used for automating

the adaptation during the search and leads to finding a solution by reflecting on the past experience stored. This is also the idea of the reactive search and intelligent optimization, with one major difference which is that in reactive search this is done for a single instance; the algorithm adapts automatically during the search and reflects on its past experience while running on a single instance. In our framework we reflect on past experience of stored cases in the case base, so our search is incremental. Similar to reactive search optimization leading to dynamic adjustment of search parameters, the feedback from the oracle in our Pref-CBR framework leads to dynamic adjustment of the search parameters during each problem-solving episode. By each given preference from the oracle, a neighborhood adjustment takes place by forming a neighborhood around the preferred solution. Throughout this feedback loop, a dynamic self-adaptation of the neighborhood takes place and the adaptation process of a solution leads to finding an optimal solution at the end of the search procedure. In reactive search and intelligent optimization, the designer for the algorithm must proactively insert modules which enable the algorithm to perform the automatic adaptation which requires the designer to have knowledge of the field of the application, while in our Pref-CBR framework what is most important is to define suitable similarity measures between problems and solutions without the necessity to be knowledgeable about the field of the application.

### 6.2.4 Selection of features

Another part of machine learning worth mentioning is the feature selection, which has been a productive field of research and development in data mining, machine learning and statistical pattern recognition, and is widely applied to many fields such as, image retrieval, genomic analysis and text categorization [18]. As in all scientific challenges, the development of models with predicting power has to start from appropriate measurements, statistics and input features. It is very important before starting to learn a parametric or nonparametric model from the examples, to consider that the input features have sufficient information to predict the outputs [139]. The removal of irrelevant and redundant information often improves the performance of machine learning algorithms. There are two common approaches for evaluation of features: a wrapper which uses the intended learning algorithm itself to evaluate the usefulness of features, or a filter which evaluates features according to heuristics based on the general characteristics of the data [145]. The feature selection can also be done using the graph

clustering approach which is based on theoretic graphs; the most relevant features are selected from the cluster for the relevant corresponding target class (in case of classification) [18]. In any case, whichever method is used for selection of features, the goal is to find the set of input features which lead to optimal output results.

As for the selection of features, in our case our similarity learning algorithms can be used to evaluate the usefulness of input features. This is basically achieved from our learning from examples algorithm for learning features of the problems; important features are emphasized and given more weight over features which do not significantly contribute to the choice for an optimal solution. This emphasis on the important features is learned by the qualitative feedback, which in turn depends on the solutions of the cases as described in detail in Chapter 4. This learning includes indirectly the connection between the problem and solution spaces, which leads to assigning importance of features in the problem space which lead to optimal output results.

## 6.3 Conclusion

In this chapter, we have listed some methodologies which relate to our Pref-CBR framework and to our work. These methodologies have been explained and some practical applications have been mentioned, for describing how these methodologies are used and applied practically. We also showed how these different methodologies are related to our Pref-CBR framework, similarities and differences between our framework and the discussed approaches. In spite of the existence of many related methodologies, there was still a good reason to develop Pref-CBR because it basically provides a means for integrating experience for solving problems without strict formalization of that experience. In other words, qualitative expert feedback is used to guide the search rather than quantitative feedback which is used by most approaches. The expert feedback embedded in our Pref-CBR framework provides a means for solving tasks for assisting people; it is not only used for classification or regression, but rather for problem-solving when there is a need for modifying a given solution to fit the requirements of the case to be solved. Other methods could be more useful though in applications where the need for precise quantitative values for solutions are seeked, such as in the field of astronomy or construction. In the next chapter we will describe and show some experiments which were implemented during our framework development. For each added part to

our framework, one or more experiments will be shown and described in detail, showing the effectiveness of our different components added within our framework.

# 7

# Experiments

In this chapter, a set of experiments will be listed in detail, showing the efficacy of the Pref-CBR framework. The first set of experiments will illustrate how the framework operates and will show how the preference-based search performs over a set of given problems. The second section will include an illustration of the effectiveness of the two approaches for learning similarity measures in the problem and solution spaces, learning by examples (using perceptron algorithm) in the problem space and the Bayesian learning approach in the solution space. The third section will include a set of illustrations showing the effect of the four different maintenance strategies for preference-based CBR, which were discussed in detail in Chapter 5. The set of experiments of the added components (learning of similarity measures and maintenance strategies) in Section 7.2 and Section 7.3 respectively, are tested in isolation to see their effect in the Pref-CBR framework. The last section will include an application for image correction which includes all the added components to the framework, showing how generic our Pref-CBR framework is, by applying it also in the image processing area. As a summary, the following illustrations are all in different fields, have different scenarios showing how our Pref-CBR framework is indeed a generic framework holding the capability of being used in different domains.

## 7.1 Pref-CBR search performance

The following experiments show the search performance of the Pref-CBR framework versus random search as a baseline for comparison. The first one is an application from

the medical domain while the second one is an application in the food domain.

### 7.1.1 Drug discovery

The function of a protein in a living organism can be modulated by ligand molecules that specifically bind to the protein surface and thereby block or enhance its biochemical activity. This is how a drug becomes effective: By docking to a protein and changing its activity, it (hopefully) interrupts a cascade of reactions that might be responsible for a disease.

The identification and selection of ligands targeting a specific protein is of high interest for de-novo drug development, and is nowadays supported by computational tools and molecular modeling techniques. Molecular docking is an *in silico* technique to screen large molecule databases for potential ligands. Using the spatial (three-dimensional) structure and physicochemical properties of proteins, it tries to identify novel ligands by estimating the binding affinity between small molecules and proteins. However, since docking results are not very reliable, they need to be controlled by human experts. This is typically done through visual inspection, i.e., by looking at the docking poses predicted by the software tool and judging whether or not a molecule is indeed a promising candidate. Needless to say, this kind of human intervention is costly. Besides, a human will normally not be able to score a docking pose in terms of a numerical (affinity) degree, whereas a comparison of two such poses can be accomplished without much difficulties. Therefore, the search for a ligand that well interacts with a target protein is a nice example of the kind of problem we have in mind.

We conducted experiments with a data set consisting of 588 proteins, which constitute the problem space $\mathbb{X}$, and 38 molecules, which correspond to the solution space $\mathbb{Y}$; this data set is an extension of the data used in [146]. For each protein/molecule pair, the data contains an affinity score (pairwise binding energy) computed by a docking tool. We make use of these scores in order to mimic a human expert, i.e., to realize our oracle: Given a protein and two candidate molecules, the oracle can provide a preference by looking at the corresponding affinity scores. As a similarity $S_X$ on problems (proteins), we used the measure that is computed by the CavBase database; this measure compares proteins in terms of the spatial and physicochemical properties of their respective binding sites [147]. For the solutions (ligands), a similarity $S_Y$ was determined based on molecular fingerprints derived from the SMILES code using a

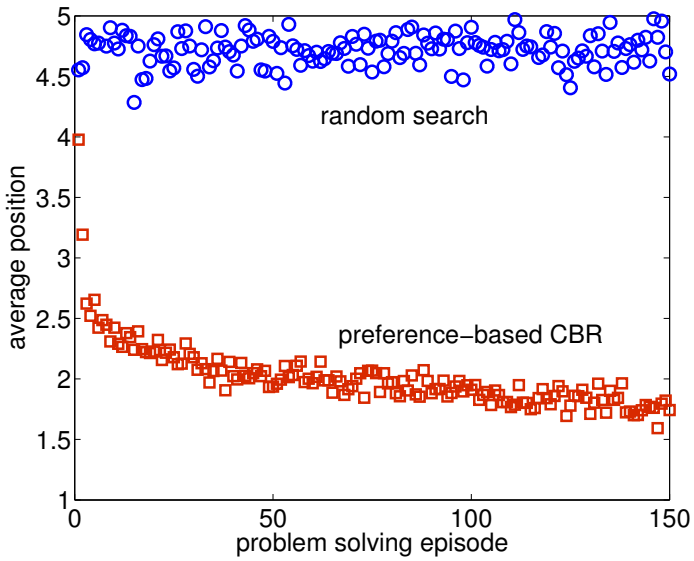


**Figure 7.1:** Average performance of Pref-CBR and random search on the drug discovery problem in the first 150 problem solving episodes.

molecular operating environment. These fingerprints were used to create a graph representation of the molecules, for which the Tanimoto similarity was determined [148]. Both similarities $S_X$ and $S_Y$ were normalized to the unit interval, and corresponding distances $\Delta_X$ and $\Delta_Y$ were defined as $1 - S_X$ and $1 - S_Y$, respectively.

We applied Algorithm 1, described in detail in Chapter 3, with $\mathbb{X}_0$ as a random order of the complete problem space $\mathbb{X}$. Since the solution space is quite small, we used a global neighborhood structure, i.e., we defined the neighborhood of a solution $\boldsymbol{y}$ as $\mathcal{N}(\boldsymbol{y}) = \mathbb{Y} \backslash \{\boldsymbol{y}\}$. As a performance $q$ of a proposed solution $\boldsymbol{y}^*$ for a problem $\boldsymbol{x}_0$ (line 21), we computed the position of this solution in the complete list of $|\mathbb{Y}| = 38$ ligands ranked by affinity to $\boldsymbol{x}_0$ (i.e., 1 would be the optimal performance). To stabilize the results and make trends more visible, the corresponding sequence of $|\mathbb{X}| = 588$ performance degrees produced by a single run of Algorithm 1 was averaged over 1000 such runs.

As a baseline to compare with, we used a search strategy in which the preference-guided selection of the next candidate solution in line 15 of Algorithm 1 is replaced by a random selection (i.e., an element from $\mathbb{Y}^{nn}$ is selected uniformly at random). Although this is a very simple strategy, it is suitable to isolate the effect of guiding the search behavior on the basis of preference information. Figure 7.1 shows the results for

parameters $K = 3$, $L = 5$, $J = 15$ in Algorithm 1 (other settings led to qualitatively similar results). As can be seen, our preference-based CBR approach shows a clear trend toward improvement from episode to episode, thanks to the accumulation and exploitation of problem solving experience. As expected, such an improvement is not visible for the random variant of the search algorithm.

### 7.1.2 Set completion

In a second experiment, we considered a set completion problem that is similar to the problem solved by the Bayesian set algorithm proposed in [149]. Given a (small) subset of items as a seed, the task is to extend this seed by successively adding (or potentially also removing) items, so as to end up with a "good" set of items. As a concrete example, imagine that items are ingredients, and itemsets correspond to (simplified) representations of cooking recipes. Then, the problem is to extend a seed like $\{\texttt{noodles}, \texttt{chicken}\}$, suggesting that a user wants a meal including noodles and chicken, to a complete and tasty recipe.

More formally, both the problem space and the solution space are now given by $\mathbb{X} = \mathbb{Y} = 2^{\mathcal{I}}$, where $\mathcal{I} = \{\iota_1, \ldots, \iota_N\}$ is a finite set of items; thus, both problems and solutions are itemsets. We define the distance measures $\Delta_X$ and $\Delta_Y$ in terms of the size of the symmetric difference $\Delta$, i.e.,

$$\Delta_X(\boldsymbol{x}, \boldsymbol{x}') = |\boldsymbol{x} \,\Delta\, \boldsymbol{x}'| = |\boldsymbol{x} \setminus \boldsymbol{x}'| + |\boldsymbol{x}' \setminus \boldsymbol{x}| \ .$$

Let $\mathbb{Y}^* \subset \mathbb{Y}$ be a set of reference solutions (e.g., recipes of tasty meals). For a $\boldsymbol{y} \in \mathbb{Y}$, define the distance to $\mathbb{Y}^*$ as

$$d(\boldsymbol{y}) = \min_{\boldsymbol{y}^* \in \mathbb{Y}^*} |\boldsymbol{y} \,\Delta\, \boldsymbol{y}^*| \ .$$

Moreover, for a problem $\boldsymbol{x} \in \mathbb{X}$, we define a preference relation on $\mathbb{Y}$ as follows: $\boldsymbol{y} \succ \boldsymbol{z}$ if either $c(\boldsymbol{y} \,|\, \boldsymbol{x}) < c(\boldsymbol{z} \,|\, \boldsymbol{x})$ or $c(\boldsymbol{y} \,|\, \boldsymbol{x}) = c(\boldsymbol{z} \,|\, \boldsymbol{x})$ and $|\boldsymbol{y}| < |\boldsymbol{z}|$, where

$$c(\boldsymbol{y} \,|\, \boldsymbol{x}) = \begin{cases} d(\boldsymbol{y}) & \text{if } \boldsymbol{y} \supseteq \boldsymbol{x} \\ \infty & \text{otherwise} \end{cases}$$

Thus, the worst solutions are those that do not fully contain the original seed. Among the proper extensions of the seed, those being closer to the reference solutions $\mathbb{Y}^*$ are preferred; if two solutions are equally close, the one with less items (i.e., the less

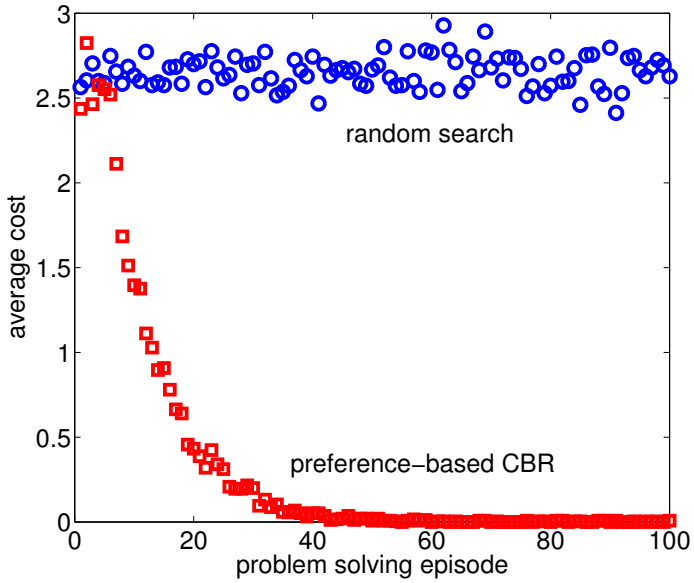


**Figure 7.2:** Average performance of Pref-CBR and random search on the set completion problem in the first 100 problem solving episodes.

expensive one) is preferred to the larger one. For a candidate solution $\boldsymbol{y}$, we define the neighborhood as the set of those itemsets that can be produced by adding or removing a single item:

$$\mathbb{Y}^{nn} = \left\{ \boldsymbol{y}' \,|\, \Delta_Y(\boldsymbol{y}, \boldsymbol{y}') = 1 \right\} \quad .$$

Finally, for a given problem $\boldsymbol{x}_0$, we define the performance of a found solution $\boldsymbol{y}^*$ in terms of $c(\boldsymbol{y}^* \,|\, \boldsymbol{x}_0)$.

We applied this setting to a database of pizzas extracted from the website **allrecipes.com**, each one characterized by a number of toppings (typically between 6 and 10). Seeds (problems) were produced at random by picking a pizza and removing all except three toppings. The task is then to complete this seed by adding toppings, so as to produce a tasty pizza (preferably one of those in the database, which plays the role of the reference set $\mathbb{Y}^*$). Again, we compared Algorithm 1 with the random search variant as a baseline. The results for parameters $K = 5$, $L = 10$, $J = 50$, shown in Figure 7.2, which are qualitatively similar to those of the previous study.

## 7.2 Learning of similarity measures

Having seen how Pref-CBR performs compared to random search, we can now see in isolation the effect of the component of learning the similarity measures on the performance. In the first subsection, we will present an illustration which shows the effect of learning similarity measures in the solution space, and we will see how the number of queries to the oracle makes a difference in the performance. The following illustration in the proceeding subsection compares the effectiveness of the learning method for the problems (perceptron) and the learning method for the solutions (Bayesian approach) and both combined, on the performance of the search. These are compared with the performance without any learning.

### 7.2.1 Wine recommendation – solution space learning

In this case study, we applied Pref-CBR to the problem of wine recommendation. The scenario is as follows: A wine merchant tries to find his best offer for a customer, i.e., that wine in his cellar the customer likes the most. For an average customer, it will be much easier to (qualitatively) compare two wines instead of rating an individual wine—this nicely fits the assumption of our framework. Thus, the merchant can offer different candidate wines to the customer (who plays the role of our oracle), which are always compared to the current favorite. For obvious reason, however, the number of such comparisons needs to be limited.

To simulate this scenario, we made use of the red wine data set from the UCI machine learning repository [150]. This data describes 4898 wines in terms of different chemical properties; here, we only used three of them, namely sulphates ($y_1$), pH ($y_2$), and total sulfur dioxide ($y_3$), which were found to have the strongest influence on preference [151]. We randomly extracted 500 wines to constitute the wines in the cellar, while 1500 other randomly extracted wines were used as queries. Thus, a query is a wine that is thought of as the ideal solution for a customer (in this example, problem space and solution space therefore have the same structure).

We defined the distance measures in terms of:

$$\Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*) = \sum_{i=1}^{k} \alpha_i \cdot \Delta_i(\boldsymbol{y}, \boldsymbol{y}^*) \ , \tag{7.1}$$

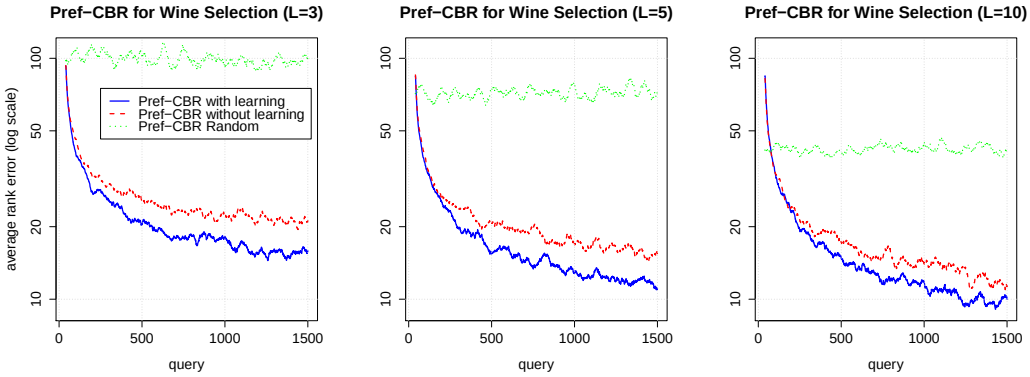


**Figure 7.3:** Evolution of the average rank error in sequential problem solving (each query gives rise to one problem solving episode) for $L = 3, 5$ and 10 queries.

with the local distances given as $\Delta_i(\boldsymbol{y}, \boldsymbol{y}^*) = |y_i - y_i^*|$, $i = 1, 2, 3$. Moreover, assuming that the different chemical properties have a different influence on taste, we defined the *ground truth* distances $\Delta_X = \Delta_Y$ by setting $\alpha_1 = 0.1$, $\alpha_2 = 0.6$, $\alpha_3 = 0.3$. However, we assume this ground truth measure, which is the target of our similarity learning method, to be unknown. Instead, the measure used in the solution space as an initial measure subject to adaptation (and in the problem space without adaptation) is the *default measure* with uniform weights $\alpha_1 = \alpha_2 = \alpha_3 = 1/3$.

We used Algorithm 2, described in detail in Chapter 4, for sequential problem solving, starting with an empty case base. Then, we applied the algorithm to the 1500 query cases one by one and monitored its performance. To measure the quality $q$ of a proposed solution $\boldsymbol{y}^\bullet$ for a problem $\boldsymbol{x}_0$ (line 22), we computed the position of this solution in the complete list of $|\mathbb{Y}| = 500$ wines in the store ranked by (ground truth) similarity to the query $\boldsymbol{x}_0 = \boldsymbol{y}^*$ (i.e., 1 would be the optimal performance). To stabilize the results and make trends more visible, the corresponding sequence of performance degrees produced by a single run of Pref-CBR Search was averaged over 100 such runs.

We compared two versions of Pref-CBR Search, namely with and without (solution) similarity adaptation. Moreover, as a baseline we also used a search strategy in which the preference-guided selection of the next candidate solution in line 15 of Algorithm 2 is replaced by a random selection (i.e., an element from $\mathbb{Y}^{nn}$ is selected uniformly at random). Although this is a very simple strategy, as we mentioned before in the experiments in the previous section, it is suitable to isolate the effect of guiding the

search behavior on the basis of preference information.

We applied our Algorithm 2 with $K = 5$, $L \in \{3, 5, 10\}$, $J = 25$; since the solution space is quite small, we used a global neighborhood structure, i.e., we defined the neighborhood of a solution $\boldsymbol{y}$ as $\mathcal{N}(\boldsymbol{y}) = \mathbb{Y} \setminus \{\boldsymbol{y}\}$. As can be seen from the results in Figure 7.3, our preference-based CBR approach shows a clear trend toward improvement from episode to episode, as opposed to the random variant of the search algorithm.

More importantly, however, similarity adaptation is clearly beneficial: Making use of the preference information gathered in the first episodes, Pref-CBR Search succeeds in learning the ground truth similarity measure, which in turn leads to better search performance and solution quality. The variant without similarity adaptation finds reasonably good solutions, too, because even the suboptimal (default) measure is guiding the search in a right direction—yet, with similarity adaptation enabled, the search becomes more effective, and the smaller the number of queries ($L$), the more pronounced the relative improvement.

### 7.2.2 Red wine recommendation – comparison of similarity measures of problems and solutions

This illustration uses the same data which was used in the above subsection, but using only the red wine data which consists of 1599 instances. We applied the Algorithm 2 to the 400 randomly extracted query cases one by one and monitored its performance. We take 50 wines to constitute the wines in the cellar, while the remaining 350 were used as queries. The measure used in the problem space and solution space as an initial measure subject to adaptation is the *default measure* with uniform weights $\alpha_1 = \alpha_2 = \alpha_3 = 1/3$.

We used Algorithm 2 for sequential problem solving, starting with an empty case base. Then, we applied the algorithm to the 350 query cases one by one and monitored its performance. To measure the quality $q$ of a proposed solution $\boldsymbol{y}^\bullet$ for a problem $\boldsymbol{x}_0$ (line 22), we computed the position of this solution in the complete list of $|\mathbb{Y}| = 50$ wines in the store ranked by (ground truth) similarity to the query $\boldsymbol{x}_0 = \boldsymbol{y}^*$ (i.e., 1 would be the optimal performance). To stabilize the results and make trends more visible, the corresponding sequence of performance degrees produced by a single run of Pref-CBR Search was averaged over 200 such runs.

We compared four versions of Pref-CBR Search, namely without similarity adaptation, with problem similarity adaptation only, with solution similarity adaptation

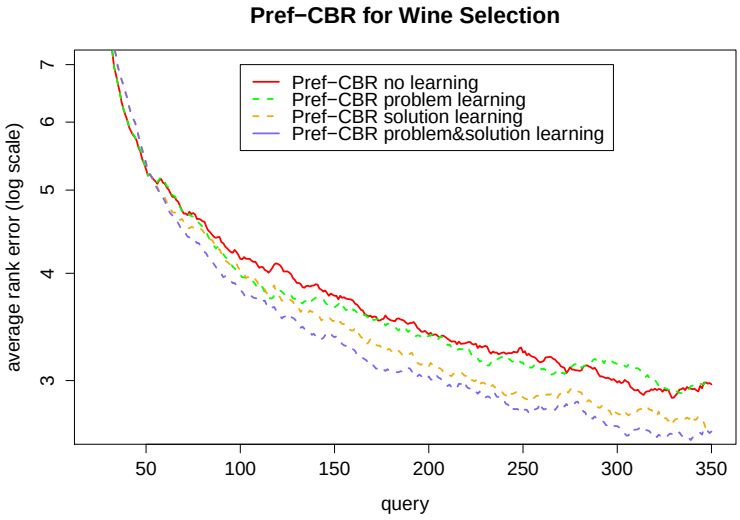


**Figure 7.4:** Comparison of performance between no metric learning, problem metric learning, solution metric learning, problem&solution metric learning.

only and with both problem and solution similarity adaptation. We applied our Algorithm 2 with $K = 5$, $L = 5$, $J = 25$; since the solution space is quite small, we used a global neighborhood structure, i.e., we defined the neighborhood of a solution $\boldsymbol{y}$ as $\mathcal{N}(\boldsymbol{y}) = \mathbb{Y} \setminus \{\boldsymbol{y}\}$. As can be seen from the results in Figure 7.4, our preference-based CBR approach shows a clear trend toward improvement from episode to episode. We can observe that applying similarity learning in the problem space shows that the search becomes more effective, similarity adaptation in the solution space shows more search improvement and clearly with both problem and solution similarity adaptation the search shows the highest performance.

## 7.3 Case base maintenance in Pref-CBR

In this section four sets of experiments will be described, two sets of experiments illustrating the inter-case maintenance methods and two sets illustrating the intra-case maintenance methods. Recalling from Chapter 5, in which detailed descriptions of the aforementioned methods are shown, the inter-case maintenance methods refer to whole deletion of cases while the intra-case maintenance methods refer to parts of cases being deleted. Both methods contain redundancy and noise deletion strategies.

We conducted an experimental study with the traveling salesman problem (TSP), i.e., with TSP instances as problems and tours as candidate solutions. Needless to say, our ambition is not to develop new state-of-the-art solvers for this NP-hard optimization problem—obviously, our completely generic problem solving framework cannot compete with specialized TSP solvers. Nevertheless, combinatorial optimization problems such as TSP provide an interesting test bed for Pref-CBR:

- In practice, such problems often need to be solved repeatedly (imagine, for example, a conveyance planning a tour every day), suggesting a reuse of previous solutions [64]; interestingly, the TSP problem has already been tackled by means of CBR by other authors [152, 153].
- The solution space $\mathbb{Y}$ is non-trivial but typically equipped with a natural structure, on which reasonable distance measures $\Delta_Y$ can be defined.
- One of the key assumptions of Pref-CBR, namely, that the optimality of a solution cannot be guaranteed, is often fulfilled—this is due to the hardness of such problems, calling for heuristic approximations.
- Nevertheless, a comparison between two candidate solutions is often possible. In TSP, for example, a preference between two tours can easily be created by computing and comparing their lengths.[1]

Another assumption of Pref-CBR, namely that a comparison is costly (and hence the number of adaptations and queries to the oracle limited), is admittedly not fulfilled in the case of TSP. Yet, one can easily imagine practically relevant generalizations of the problem for which this assumption applies. For example, suppose we replace a precise evaluation criterion such as *length* of a tour by a more "soft" criterion such as *comfort* or *convenience*. Then, to compare two candidates, it may indeed be necessary to practically try both of them (e.g., to walk a hiking tour), which might be time-consuming and involve input of a human expert (playing the role of the "oracle" then). In such cases, comparing two candidates qualitatively may also be simpler than rating them individually.

[1] Actually, we could even create more than a *qualitative* preference, because the numerical values of the solutions (lengths of the tours) are known as well. This is indeed additional information we are not exploiting in this application

### 7.3.1 Setting

The components of our Pref-CBR setting are specified as follows:

- The problem space $\mathbb{X}$ for the first experiment is the set of all subsets $\boldsymbol{x} \subset \mathcal{X}$ of size $|\boldsymbol{x}| = 10$. The problem space $\mathbb{X}$ for the second experiment is the set of all subsets $\boldsymbol{x} \subset \mathcal{X}$ of size $|\boldsymbol{x}| = 20$, where $\mathcal{X} \subset \mathbb{R}^2$ is a randomly created reference set of 25 points on the plane for the first experiment and 50 points on the plane for the second experiment; each point can be thought of as the location of a city.

- The distance $\Delta_X(\boldsymbol{x}, \boldsymbol{x}')$ between two problems is defined in terms of the sum of pairwise squared distances between cities of two instances, subsequent to an optimal assignment of the points that is obtained by solving the linear assignment problem [154] with Euclidean distance as a cost measure.

- Solutions are represented as permutations specifying the order of cities/points in a tour. Thus, $\mathbb{Y}$ is the set of all permutations of $\{1, \ldots, 10\}$ for 10 cities or of $\{1, \ldots, 20\}$ for 20 cities. This space is equipped with a local neighborhood structure by connecting each solution $\boldsymbol{y}$ with 200 "perturbations" of this solution, each of which is obtained by randomly switching the position of a small number (2, 4 or 6) of points.

- To define the distance $\Delta_Y(\boldsymbol{y}, \boldsymbol{y}')$ between two solutions, each solution is first mapped to a feature vector with the coordinates of the cities in the specified order of the permutation. Then, the corresponding feature vectors are compared in terms of their Euclidean distance.

- The parameters of Pref-CBR were set as follows: number of nearest neighbors $K = 15$, number of adaptation steps $L = 10$ for instances of 10 cities and $L = 20$ for instances of 20 cities.

### 7.3.2 Inter-case maintenance methods

In our inter-case maintenance experimental study, we compared the Pref-CBR search without case base maintenance, with inter-case redundancy and inter-case noise maintenance methods, for instances of 10 cities and 20 cities. As additional baselines, we included a random case deletion (RCD) policy, which removes each newly observed case

with a fixed probability (RCD) of 1/3. We generated a sequence of 300 instances of the TSP problem (using the "tspmeta" library in R), giving rise to the same number of problem solving episodes. Each time a solution has been produced, we measure performance by computing the ratio between the corresponding tour length and the optimal tour length found by the "cheapest_insertion" TSP solver. Since the sequence of performance values thus produced is rather noisy, we average over a larger number of repetitions of this experiment to produce smoother curves.

These curves are shown in Figure 7.5 for TSP instances of 10 cities, both for Pref-CBR without maintenance and Pref-CBR with maintenance (inter-case redundancy and inter-case noise) with values of the parameter $v = 4$ for the redundancy maintenance, and $v = 7$ for the noise maintenance (while the threshold $t$ was fixed to 1). Moreover, the evolution of the size of the case base is also shown in the plot on the right. As can be seen, the desired effect is indeed achieved: The size of the case base is reduced while performance is maintained (in contrast to the random deletion policy). Moreover, by increasing the value of $v$, the stronger the tendency to delete cases. Thus, this parameter can be used to control the size of the case base. The curves in Figure 7.6 for TSP instances of 20 cities, have parameter $v = 8.5$ for redundancy maintenance and $v = 15.5$ for noise maintenance, and also $t = 1$. The evolution of the case base size is illustrated in the plot on the right. Since case base maintenance starts after a pre-specified number of solved problems, the parameter $v$ could be approximated. After performing some experiments, we can conclude for this example that $v$ could be valued according to the average distance between solutions of solved problems. For redundancy, $v \simeq \frac{3}{4}\bar{d}_s$, where $\bar{d}_s$ is the average distance between solutions. For noise, $v \simeq \frac{3}{2}\bar{d}_s$.

### 7.3.3 Intra-case maintenance methods

In our intra-case maintenance experimental study, we compared Pref-CBR search without case base maintenance with intra-case redundancy and intra-case noise maintenance methods, for instances of 10 cities and 20 cities, with same data used in the previous section. As additional baselines, we included a random preferences deletion policy (RPD), which removes individual preferences (RPD) with the a fixed probability of 1/3.

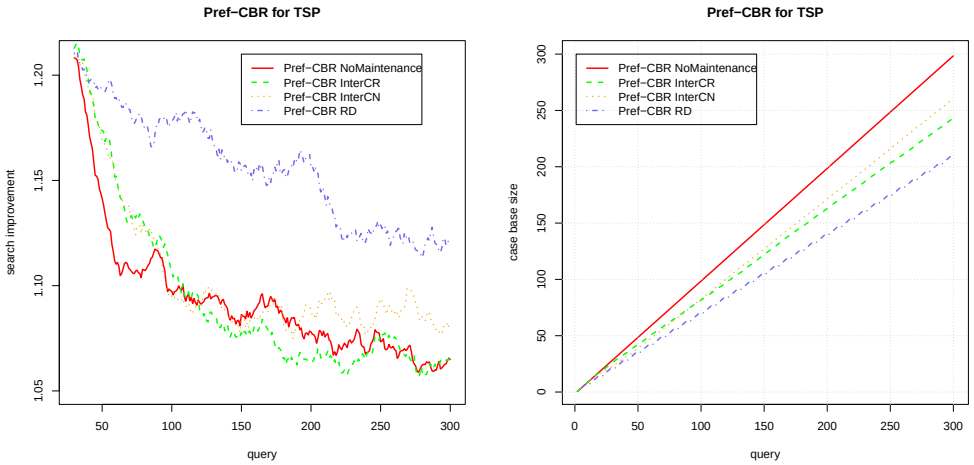


**Figure 7.5:** Pref-CBR search with and without inter-case maintenance methods of TSP (10cities) data and case base size.

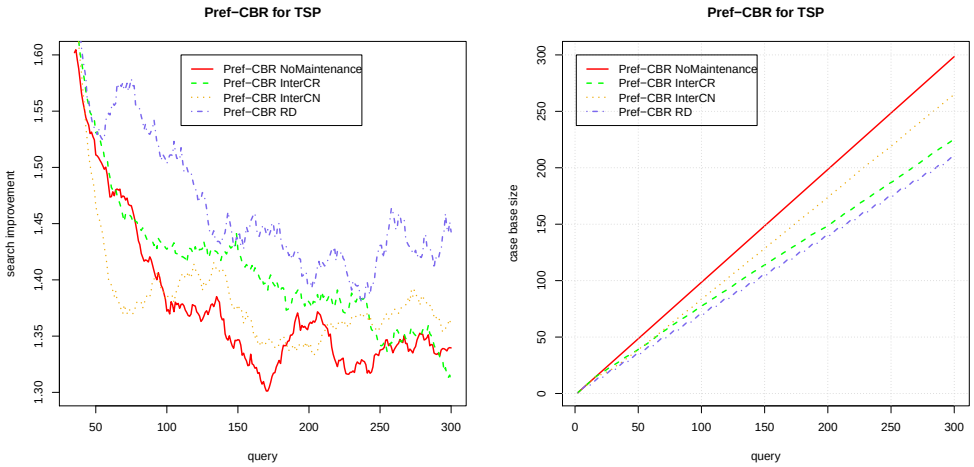


**Figure 7.6:** Pref-CBR search with and without inter-case maintenance methods of TSP (20cities) data and case base size.

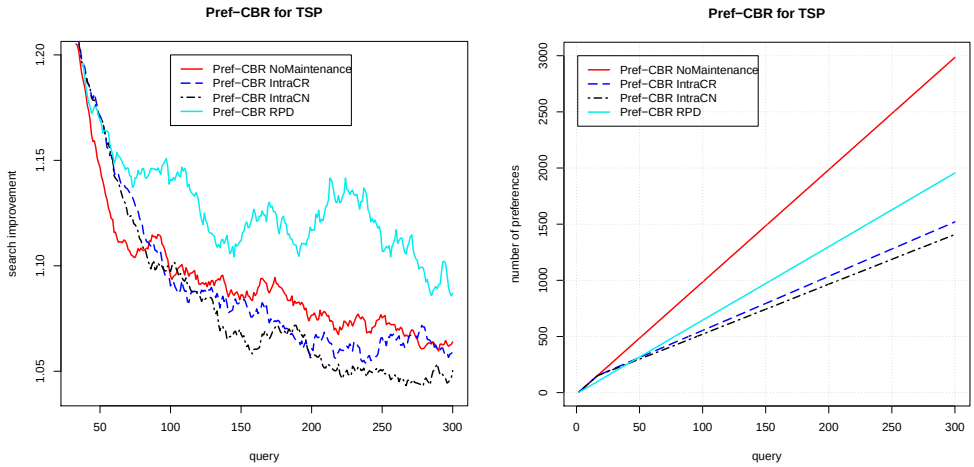


**Figure 7.7:** Pref-CBR search with and without intra-case maintenance methods of TSP (10cities) data and number of preferences in case base.

These curves are shown in Figure 7.7 for TSP instances of 10 cities, both for Pref-CBR without maintenance and Pref-CBR with maintenance (intra-case redundancy and intra-case noise). Moreover, the plot on the right shows the evolution of the size of the case base (in terms of the number of preferences). The original number of preferences should be equal to the number of cases times the number of queries to oracle, which is 3000 for instances of 10 cities and 6000 for instances of 20 cities. As can be seen, the size of the case base is significantly reduced, while performance is maintained (in contrast to the random preferences deletion policy). We can also observe that the size is reduced almost to half the case base size using the intra-case redundancy and noise maintenance methods, while deleting much less preferences randomly affects the quality of performance.

The curves in Figure 7.8 for TSP instances of 20 cities, illustrate the search performance and the evolution of the number of preferences in the case base is shown in the right plot.

We can conclude from the conducted experiments, that the inter-case redundancy methods maintain the performance and increase the efficiency of the case base. We can observe from the results of the experiments also, that our intra-case maintenance methods (redundancy and noise) actually improve the performance as well as increase the efficiency of the case base. We can therefore take advantage of our Pref-CBR

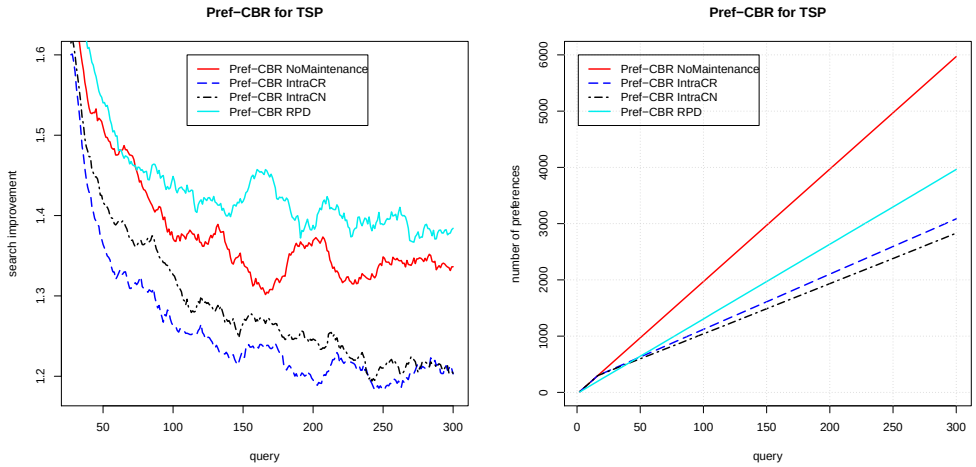


**Figure 7.8:** Pref-CBR search with and without intra-case maintenance methods of TSP (20cities) data and number of preferences in case base.

strategy of storing the preferences over solutions and instead of deleting whole cases, we can delete some preferences which are redundant or noisy, leaving more information in the case base to be used for the search. We cannot concretely generalize this result, but for the TSP data, this can clearly be observed.

## 7.4 Image improvement application

The experiment in this section is a very interesting application in the image processing domain. The illustration presented shows how nicely the Pref-CBR framework operates, even in such a complex field as image processing. We conducted an experiment with a data set consisting of 300 images (96x96 pixels), which constitute the problem space $\mathbb{X}$, and a directed set of filters, which correspond to the solution space $\mathbb{Y}$; this data set has been collected from an image recognition benchmark data set [155]. The data set contains images from different categories: airplanes, automobiles, birds, cats, deer, dogs, horses, ships and trucks. The problems are a set of distorted images and the solutions are directed sets (sequences) of filters. A sequence of filters, should when applied to the distorted image, yield an improved image (hopefully as close as possible to the original undistorted image). The values of the parameters of the aforementioned directed set of filters which improve the images, are developed during the Pref-CBR search process.

The images have been distorted by four different types of filters: brightness, blur or sharpen, gamma, and contrast. The filters' parameters varies between -1 and 1 (zero being neutral setting, meaning that a specific filter is not applied). A Gaussian distribution was applied randomly on each of the parameters for each image. Thus, parametrized filters were applied on the original images to get the distorted (query) images as our problems. These filter parameters are stored for later evaluation. It is to be noted that the Gaussian distribution for random filters which were applied for distorting the images, was clipped between the values of $\boldsymbol{y} \in [-1, 1]^4$, for the necessity of distorting an image while still maintaining some of its characteristics intact for being able to improve it. If the picture is completely distorted and none of its characteristics remain, it would be impossible to improve the image.

It is noteworthy to mention the dependencies between the different filters, which in turn were applied for distorting the images to create our problems, and would be applied as solutions for correcting or improving the distorted images later on in the search. The blur or sharpen filter and the three others are independent of each other. On the contrary, there is a dependency between the contrast, gamma and brightness filters. The contrast filter is connected to the arithmetic '$\times$' (multiplication) operator, gamma correction to the '$\wedge$' (power) operator and brightness to the '$+$' (addition) operator; these arithmetic operators being applied on the pixel values. We can then acknowledge that for a certain order of an application of filters for distorting the images, it would make sense to apply the filters in the reverse order for correcting these images. Although this may seem to be a simple task, it is surely otherwise. Generally, for an image X, applying the filters of gamma, contrast and brightness, we would have such an equation: $U = (X^{(c+1)})(d + 1) + e$ where neutral parameter settings would be a vector of zeros. To recover image X again, the filters would be applied in the reverse order such as: $U = ((X + e)(d + 1))^{(c+1)}$

The order of the operation makes a difference and the challenge here is that image X is not only a single numeric value, but rather an image containing 96x96x3 RGB values, that is the addition, multiplication and power operators, are applied to each component. In other words, using these operators on the parameters affect the whole set of 96x96x3 RGB values. The Pref-CBR constituents are listed as follows:

- The problems consist of distorted images in feature representation. The 8 feature

groups for each image are: entropy, correlation, contrast, mean, homogeneity, variance, dissimilarity, and color. Each of the aforementioned groups, except for the color group, was calculated by the R package glcm (grey-level co-occurrence matrices) [156], and contains 10 components with values for the 5 percent, 15 percent, etc. to 95 percent quantiles. The "color" features' group is a calculation of 3x3 local patches of the correlation between the red, green, and blue color channels of an image. The range of values within each feature group was adjusted between 0 and 1 across all images, where $\mathbb{X} = [0, 1]^{80}$. The $S_X$ on problems is calculated by the Euclidean distance between the extracted feature vectors of the images (80 features).

- The solution space consists of filter parameter settings, as indicated above. The goal is to find a directed set of filters with values to improve the quality of the query image, according to the oracle where $\mathbb{Y} = [-1, 1]^4$. $S_Y$ between the solutions is measured in terms of the Euclidean distance between the parameters of the filters (4 parameters).

- The oracle represents the human in the loop, it generates a preference of the "better-looking" image. Currently for the practicality of the application, the preference is given based on a locality improved distance between a query image having the current filter settings applied (current solution), and the original image from the database. This "locality improved distance" between two images is measured by dividing each 96x96 image to local 3x3 chunks, and measuring the distance between each of these chunks of two images. We use this distance measure instead of measuring distance between whole images, to resemble more closely the retina of a human eye. It could be similar to the visual perception of a human in the loop, comparing two images and choosing the one which is visually more appealing (the one which is more similar to the original image). Our choice to use this specific distance measure for the oracle, is its consideration to the "neighbor information" of pixels.

- The quality evaluation is measured by the percent images difference between the improved distorted query image, and the original image [157]. This measure is commonly used for image comparisons.

In our experiment, we defined the neighborhood $\mathcal{N}$ of a solution $\boldsymbol{y}$ as a spherical structure (a three-dimensional neighborhood with a fixed radius) around the solution (center of the sphere). We applied Algorithm 1 with $K = 15$, $L = 12$ and $J = 180$.

The performance plot is shown in Figure 7.9, where we can observe the comparison of performances of the Pref-CBR search, the random search, the Pref-CBR search with learning similarity metrics of problem and solution spaces, and Pref-CBR with learning and intra-case redundancy maintenance. It is clearly seen that the Pref-CBR search shows a clear trend of improvement compared to the random search. We can also see the effect of learning the similarity measures in the plot, where performance is greatly further improved. Finally we can see the curve for the performance with both components of learning the similarity metrics as well as applying one of our maintenance strategies. We can distincly notice that the quality of the performance is maintained, in addition we were able to reduce the case base to approximately half of its size, thanks to our intra-case redundancy maintenance strategy. In Figure 7.10, we can see how the case base size is reduced throughout the search procedure.

The effect of learning can be nicely and concretely seen in Figure 7.11, Figure 7.12 and Figure 7.13. These figures include images for different stages during the learning (after 50 stored cases, 100, etc to 300). We can observe that as the search progresses, and more cases are stored in the case base, the solutions keep improving.

Figure 7.14, Figure 7.15, Figure 7.16, Figure 7.17 and Figure 7.18 show some examples of the progress of the correction of the images throughout the loop of queries to the oracle. We can see how the images develop from the distorted image, until reaching a solution at the end of the "queries to the oracle" loop. However, the random filters applied might exceed the color range, leading to loss of information (such as white or black patches) and these could not be completely recovered in detail.

It can be seen that the image correction application nicely shows the benefit of our Pref-CBR framework. The additional benefit of similarity metrics learning and the benefit of CBM, leading to increased efficiency, are also clearly observed. This application nicely illustrates practically the efficacy of the whole methodology of our Pref-CBR framework.

In this chapter it was clearly shown how the Pref-CBR problem-solving framework performs within different application domains. We have also shown how each added element in the framework, such as the learning of similarity measures components and

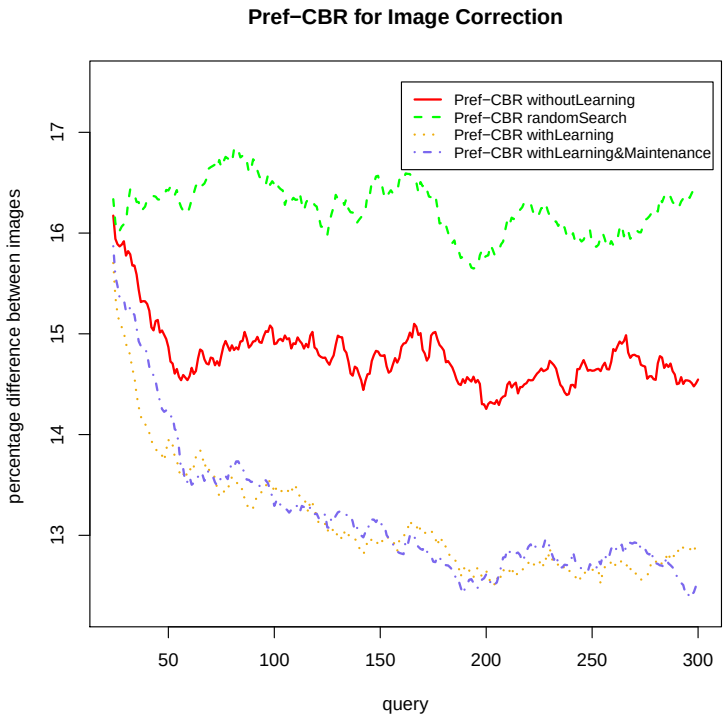


**Figure 7.9:** Performance showing different curves of average percentage difference between images.

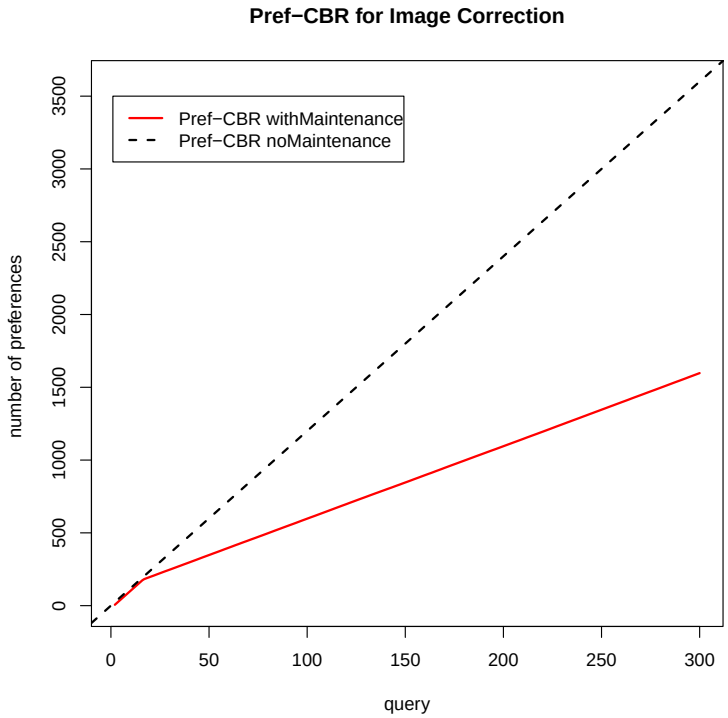


**Figure 7.10:** Case base size

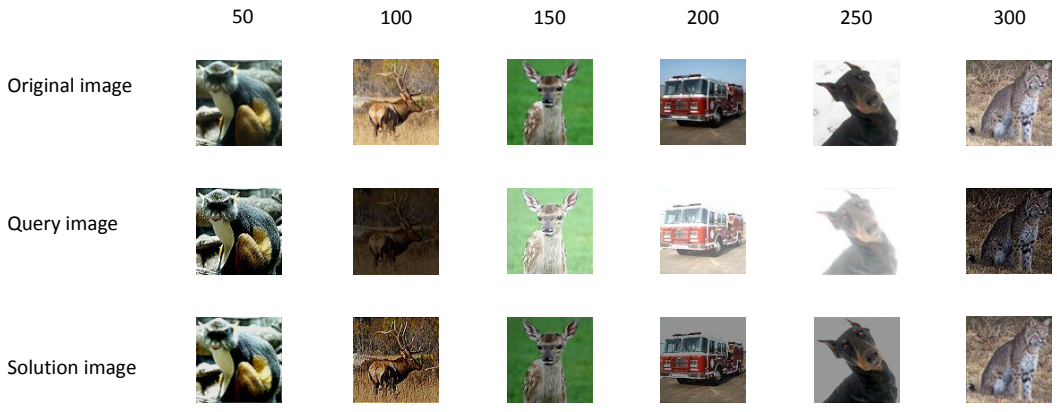


**Figure 7.11:** Comparison between original, query, and final image after applying solution (set of filters) at different intervals of case base size.

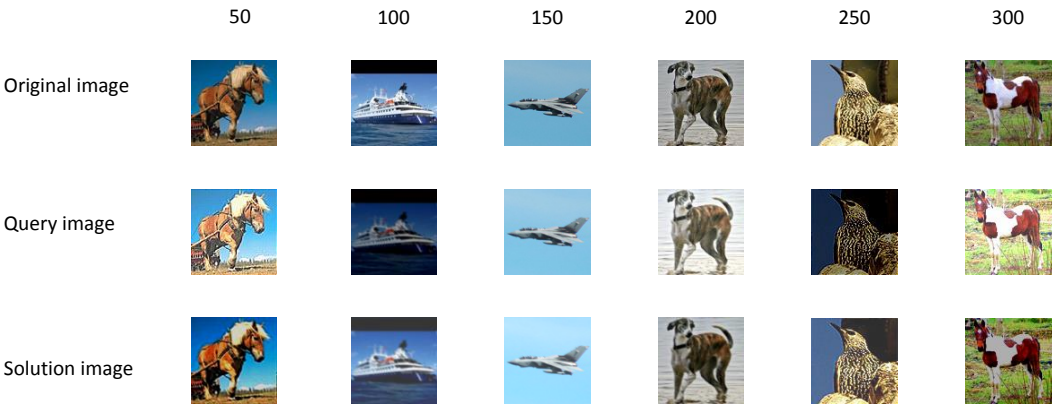


**Figure 7.12:** Comparison between original, query, and final image after applying solution (set of filters) at different intervals of case base size.

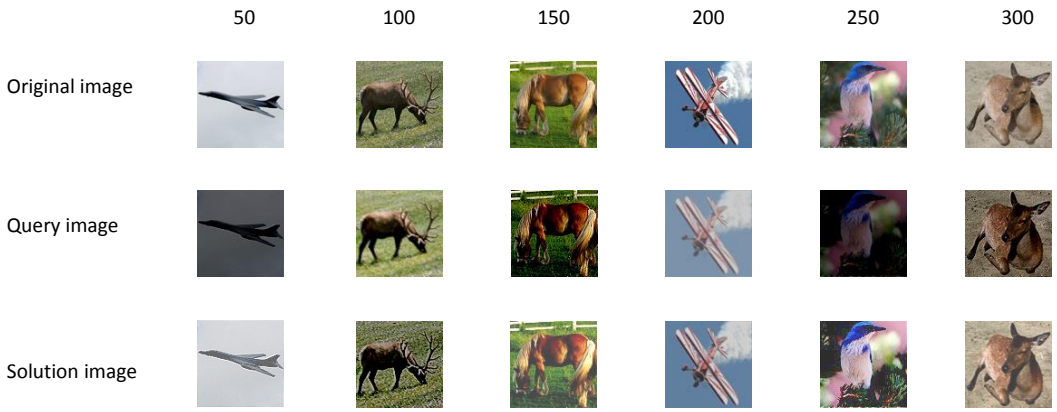


**Figure 7.13:** Comparison between original, query, and final image after applying solution (set of filters) at different intervals of case base size.

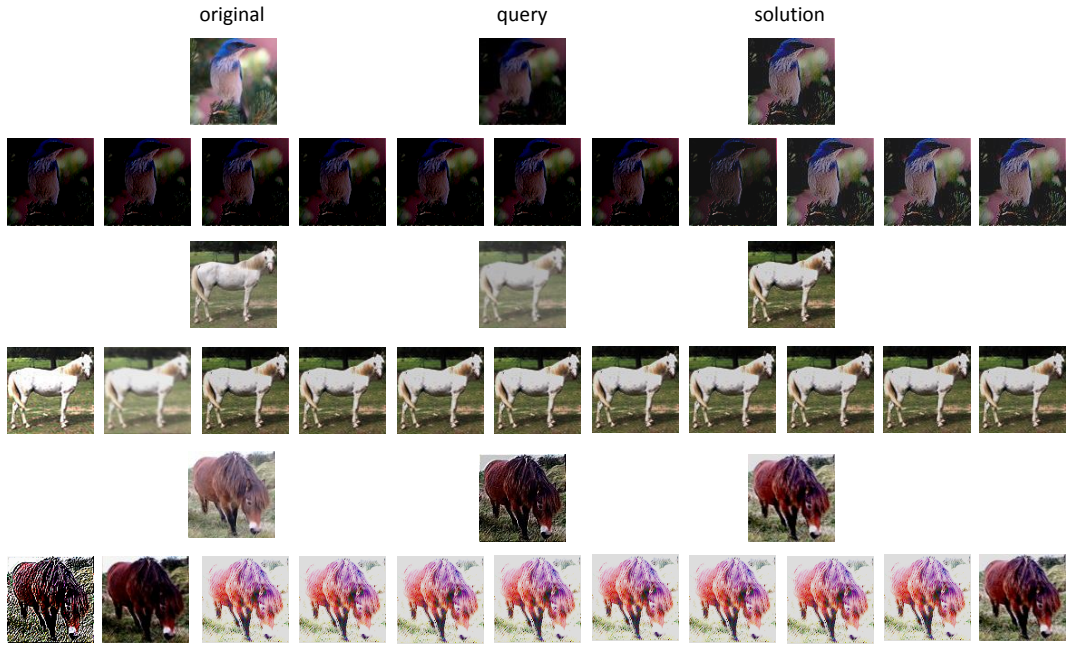


**Figure 7.14:** Progress of image improvement.

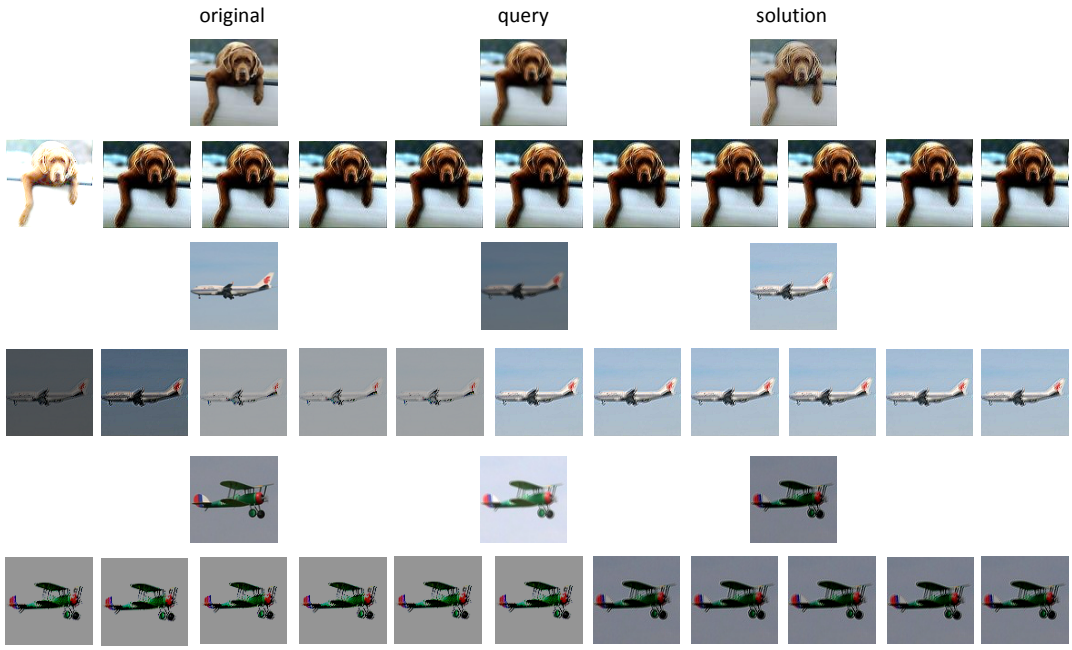


**Figure 7.15:** Progress of image improvement.


**Figure 7.16:** Progress of image improvement.

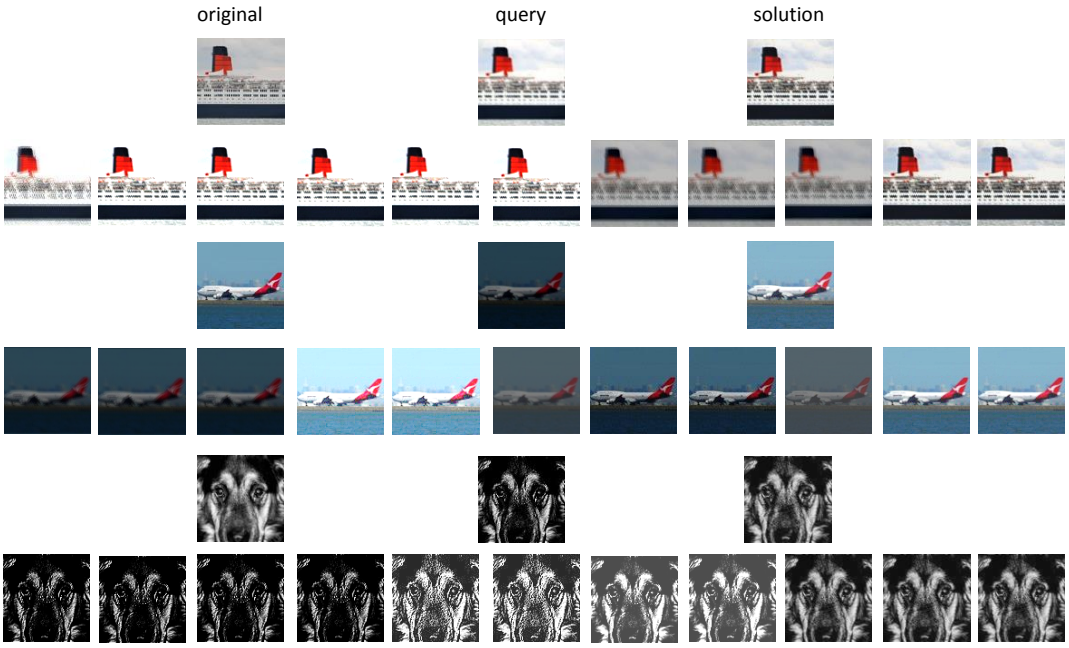


**Figure 7.17:** Progress of image improvement.


**Figure 7.18:** Progress of image improvement.

the different case base maintenance components, have also added to the optimization and efficiency of the framework. We have implemented different case studies within the medical domain, food domain, combinatorial optimization domain and the image processing domain. We have shown how our Pref-CBR framework is flexible and generic for usage in different setups and showed also how it can be used in environments where conventional CBR could not be used. The Pref-CBR problem-solving framework is based on sound theoretical methodologies, where the inference procedure uses mathematical and statistical methods from the machine learning field. The flexibility and the genericness of our framework, as well as its formulation on sound theoretical methods, in our opinion, makes it attractive and reliable to be used in different setups where conventional CBR could not be used.

# 8

# Conclusions and Outlook

In the work in this thesis, we have presented a general framework for CBR in which experience is represented in the form of contextualized preferences. These preferences are used to direct an adaptive problem solving process that is formalized as a search procedure. This kind of preference-based CBR is an interesting alternative to conventional CBR whenever solution quality is a matter of degree, and feedback is only provided in an indirect or qualitative way. We have highlighted the differences between conventional CBR and preference-based CBR, and we pointed out when Pref-CBR would be advantageous to use in real world applications.

We have described the preference-based knowledge representation; mainly that experiences of the form "solution $\boldsymbol{y}$ (optimally) solves problem $\boldsymbol{x}$" are replaced by weaker information of the form "$\boldsymbol{y}$ is better (more preferred) than $\boldsymbol{z}$ as a solution for $\boldsymbol{x}$", that is, by a preference between two solutions *contextualized* by a problem $\boldsymbol{x}$. We have explained in detail the formal setting of the framework, the case-based inference as a probability estimation, as well as how our CBR preference-guided search is performed. We presented some illustrations in the first section of Chapter 7, which show the effectiveness of our generic framework.

For further optimization of our framework, we added two extensions to it. One extension is the adaptation of our distance measures in the solution and problem spaces, and the second extension is having effective methods for case base maintenance for ensuring efficiency of the case base.

In Chapter 4, we proposed two methods for learning similarity measures: the Bayesian approach for learning metrics of the solution space, and the perceptron learn-

ing algorithm for learning metrics of the problem space. We explained in detail how the learning methods are integrated in our framework, as well as how the information gained by each problem-solving episode is used for the continuous updating of the learned metrics. The learning of the similarity measures leads to an increased performance throughout the search process, and this is clearly illustrated in the application illustrated in the second section of Chapter 7.

As we further gained knowledge of how the Pref-CBR framework operates, we came to the realization of the demanding necessity of maintaining the case base. The more the case base grows, the number of preferences may become extremely large, thus hindering the efficiency of the CBR system. The need for case base maintenance became very clear to us, and accordingly we developed four CBM strategies which specifically suit our Pref-CBR framework. We explained how the likelihood function can give us hints on defining stored preferences which are noisy or redundant. We developed two main groups of CBM strategies which consist of either deleting whole cases (inter-case maintenance) or partial case deletion (intra-case maintenance). The latter deletion method is concerned with deletion of only some preferences, rather than whole cases. The effectiveness of these approaches was illustrated in a case study of the traveling salesman problem, described in detail in the third section of Chapter 7.

Finally, we discussed some methodologies which are related to our Pref-CBR framework, and we highlighted these relations. To summarize our work and integrate the extensions of our framework, and to show the benefit of each added component, we chose to implement the "Image Improvement Application". This application shows how Pref-CBR works even in such a complex field as image processing. In addition we also included in the application the idea of having a human in the loop.

For future work, there are some issues which could be further investigated. We have done some work on adaptive neighborhoods, which have been briefly mentioned in the traveling salesman problem case study, where we created neighborhoods of switching two, four or six cities depending on the performance. We have also applied the idea of adaptive neighborhoods in some other work, where neighborhoods would be changing (size or radius) according to the quality of the performance, thus continuously improving the performance during the search procedure. For example, number of elements in the neighborhood would be adaptive (increase or decrease relative to the performance), or the radius of the neighborhood formed around an initial solution would also be adaptive,

depending on the application. This is an interesting addition to the framework which could be further investigated.

Another aspect which could also be tested is the idea of forming intelligent neighborhoods around the initial solution. Some machine learning methods could be used for intelligently forming a good neighborhood (focusing the search direction based on search history), instead of a random one (searching isotropically in all directions). In addition, the idea of using preferences for guiding the search process as we do in our framework, could be extended from heuristic search to more sophisticated search methods. Currently our maximum likelihood approach probes isotropically around a current solution to perform greedy hill-climbing search in the solution space. We could for example not limit the search to one greedy pick, but to several candidates using beam search. We could also try using local simulated annealing; stay with a certain probability at the current solution for generating better options from there, even if another one from the neighborhood might seem better.

One more thing which would be interesting to continue to work on, is the possibility of using another approach for learning similarity measures in the solution space which would overcome the limitation of its usage with a few number of attributes. In our framework, we use the Bayesian method which is elegant and complements our framework, where we create discretizations of the continuous domain and we compute the posterior numerically. The way the posterior is computed puts a limit to the number of features in the solution space, if there are many features the computation gets to be very expensive. The reason for numerically computing the posterior is that there does not seem to be a conjugate family for our current model. It would be interesting to test another learning approach to overcome this limitation, maybe by using for example point estimation, where we can adapt the metric used in the maximum likelihood to maximize the likelihood by directly using the feedback of the oracle.

# References


[1] Ralph Bergmann, Klaus-Dieter Althoff, Mirjam Minor, Meike Reichle, and Kerstin Bach. **Case-Based Reasoning – Introduction and Recent Developments**. *KI - Künstliche Intelligenz, German Journal on Artificial Intelligence - Organ des Fachbereiches "Künstliche Intelligenz" der Gesellschaft für Informatik e.V. (KI)*, **23**(1):5–11, 1 2009. 1

[2] Simon Shiu and Sankar K. Pal. *Foundations of Soft Case-Based Reasoning*. John Wiley & Sons, 2004. 1

[3] Janet L Kolodner. **An Introduction to Case-Baseased Reasoning**. *Artificial Intelligence Review*, **6**(1):3–34, 1992. 1

[4] Padraig Cunningham, Donal Doyle, and John Loughrey. **An Evaluation of the Usefulness of Case-Based Explanation**. In *Proceedings of the Fifth International Conference on Case-Based Reasoning*, pages 122–130. Springer, 2003. 1

[5] Robert H Tenback. **A Comparison of Similarity Measures for Case-Based Reasoning**. *Utrech University*, 1994. 2

[6] Ramon Lopez De Mantaras, David McSherry, Derek Bridge, David Leake, Barry Smyth, Susan Craw, Boi Faltings, Mary Lou Maher, MICHAEL T COX, Kenneth Forbus, et al. **Retrieval, Reuse, Revision and Retention in Case-Based Reasoning**. *The Knowledge Engineering Review*, **20**(03):215–240, 2005. 2, 11

[7] Agnar Aamodt and Enric Plaza. **Case-based Reasoning; Foundational Issues, Methodological Variations, and System Approaches**. *AI Communications*, **7**(1):39–59, 1994. 2, 9, 10

[8] Roger C Schank and Robert P Abelson. *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. L. Erlbaum, Hillsdale, NJ, 1977. 2

[9] Ralph Bergmann. *Experience Management: Foundations, Development Methodology, and Internet-based Applications*. Springer-Verlag, Berlin, Heidelberg, 2002. 2

[10] Alec Holt, Isabelle Bichindaritz, Rainer Schmidt, and Petra Perner. **Medical Applications in Case-Based Reasoning**. *The Knowledge Engineering Review*, **20**(03):289–292, 2005. 2

[11] Isabelle Bichindaritz. **Case-Based Reasoning in the Health Sciences: Why It Matters for the Health Sciences and for CBR**. In Klaus-Dieter Althoff, Ralph Bergmann, Mirjam Minor, and Alexandre Hanft, editors, *Advances in Case-Based Reasoning*, **5239** of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2008. 2

[12] Eyke Hüllermeier and Patrice Schlegel. **Preference-Based CBR: First Steps toward a Methodological Framework**. In Ashwin Ram and Nirmalie Wiratunga, editors, *Case-Based Reasoning Research and Development*, **6880** of *Lecture Notes in Computer Science*, pages 77–91. Springer Berlin Heidelberg, 2011. 4, 5, 26, 29, 39

[13] Jon Doyle. **Prospects for Preferences**. *Computational Intelligence*, **20**(2):111–136, 2004. 4

[14] Judy Goldsmith and Ulrich Junker. **Preference Handling for Artificial Intelligence**. *AI Magazine*, **29**(4):9–12, 2008. 4

[15] Carmel Domshlak, Eyke Hüllermeier, Souhila Kaci, and Henri Prade. **Preferences in AI: An Overview**. *Artificial Intelligence*, **175**(78):1037 – 1052, 2011. Representing, Processing, and Learning Preferences: Theoretical and Practical Challenges. 4

[16] Ronen Brafman and Carmel Domshlak. **Preference Handling-An Introductory Tutorial**. *AI Magazine*, **30**(1):58, 2009. 4

[17] Weiwei Cheng and Eyke Hüllermeier. **Learning Similarity Functions from Qualitative Feedback**. In *Advances in Case-Based Reasoning*, pages 120–134. Springer, 2008. 5, 47, 56

[18] Harshali D Gangurde. **Article: Feature Selection using Clustering Approach for Big Data**. *IJCA Proceedings on Innovations and Trends in Computer and Communication Engineering*, **ITCCE**(4):1–3, December 2014. Full text available. 6, 95, 96

[19] Vahid Jalali and David Leake. **Adaptation-Guided Case Base Maintenance**. In *Proc. AAAI, National Conference on Artificial Intelligence*, 2014. 6, 63, 65

[20] David B Leake. **CBR in Context: The Present and Future**. *Case-Based Reasoning, Experiences, Lessons & Future Directions*, pages 1–30, 1996. 9, 10, 14, 15, 21, 22

[21] Eyke Hüllermeier. *Case-Based Approximate Reasoning*, **44**. Springer Science & Business Media, 2007. 9

[22] Stephen Slade. **Case-based Reasoning: A Research Paradigm**. *AI Magazine*, **12**(1):42, 1991. 10, 12, 14, 23

[23] Roger C Schank. *Dynamic memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, 1983. 10

[24] Enrico Blanzieri and Francesco Ricci. **Probability Based Metrics for Nearest Neighbor Classification and Case-Based Reasoning**. In *Case-Based Reasoning Research and Development*, pages 14–28. Springer, 1999. 11

[25] Ian Watson. **An Introduction to Case-Based Reasoning**. In *Progress in Case-Based Reasoning*, pages 1–16. Springer, 1995. 11

[26] Gavin Finnie and Zhaohao Sun. **Similarity and Metrics in Case-Based Reasoning**. *International Journal of Intelligent Systems*, **17**(3):273–287, 2002. 11

[27] T Warren Liao, Zhiming Zhang, and Claude R Mount. **Similarity Measures for Retrieval in Case-Based Reasoning Systems**. *Applied Artificial Intelligence*, **12**(4):267–288, 1998. 11

[28] Edmund K. Burke, Bart MacCarthy, Sanja Petrovic, and Rong Qu. **Structured Cases in Case-Based Reasoning: Re-Using and Adapting Cases for Time-Tabling Problems**. *Knowledge-Based Systems*, **13**(2):159–165, 2000. 11

[29] Niloofar Arshadi and Kambiz Badie. **A Compositional Approach to Solution Adaptation in Case-Based Reasoning and its Application to Tutoring Library**. In *Proceedings of 8th German Workshop on Case-Based Reasoning. Lammerbuckel*, 2000. 11

[30] Thomas R Roth-Berghofer. **Explanations and Case-Based Reasoning: Foundational Issues**. In *Advances in Case-Based Reasoning*, pages 389–403. Springer, 2004. 14

[31] Ralph Bergmann and Wolfgang Wilke. **Towards a New Formal Model of Transformational Adaptation in Case-Based Reasoning**. In *ECAI*, pages 53–57, 1998. 15, 34

[32] David W Aha and Héctor Muñoz-Avila. **Introduction: Interactive Case-Based Reasoning**. *Applied Intelligence*, **14**(1):7–8, 2001. 15

[33] David W Aha, Leonard A Breslow, and Héctor Muñoz-Avila. **Conversational Case-Based Reasoning**. *Applied Intelligence*, **14**(1):9–32, 2001. 15, 16

[34] Hideo Shimazu. **ExpertClerk: A Conversational Case-Based Reasoning Tool for Developing Salesclerk Agents in E-Commerce Webshops**. *Artificial Intelligence Review*, **18**(3-4):223–244, 2002. 15, 16

[35] David McSherry. **Conversational Case-Based Reasoning in Medical Decision Making**. *Artificial Intelligence in Medicine*, **52**(2):59 – 66, 2011. Artificial Intelligence in Medicine {AIME} 2009. 15, 16, 17

[36] David Mcsherry. **Increasing Dialogue Efficiency in Case-Based Reasoning Without Loss of Solution Quality**. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 121–126. Morgan Kaufmann, 2003. 16

[37] Barry Smyth and Lorraine McGinty. **The Power of Suggestion**. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 127–132, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc. 17

[38] Lorraine Mc Ginty and Barry Smyth. **Comparison-Based Recommendation**. In *Lecture Notes in Computer Science*, page 2002, 2002. 17, 18

[39] Maria Salam, James Reilly, Lorraine McGinty, and Barry Smyth. **Knowledge Discovery from User Preferences in Conversational Recommendation**. In AlpioMrio Jorge, Lus Torgo, Pavel Brazdil, Rui Camacho, and Joo Gama, editors, *Knowledge Discovery in Databases: PKDD 2005*, **3721** of *Lecture Notes in Computer Science*, pages 228–239. Springer Berlin Heidelberg, 2005. 18

[40] Michael M Richter and Rosina O Weber. **Case-Based Reasoning**. *A Textbook*, page 546, 2013. 18, 19, 20, 21

[41] Norbert Gronau and Frank Laskowski. **Using Case-Based Reasoning to Improve Information Retrieval in Knowledge Management Systems**. In *Advances in Web Intelligence*, pages 94–102. Springer, 2003. 19

[42] Selma Limam, HA Reijers, and Farhi Marir. **Case-Based Reasoning as a Technique for Knowledge Management in Business Process Redesign**. *Electronic Journal on Knowledge Management*, **1**(2):89, 2003. 20

[43] Ian Watson. **Knowledge Management and Case-Based Reasoning: A Perfect Match?.** In *FLAIRS Conference*, pages 118–122, 2001. 20

[44] Atilim Güneş Baydin, Ramon López de Mántaras, Simeon Simoff, and Carles Sierra. **CBR with Commonsense Reasoning and Structure Mapping: An Application to Mediation**. In *Case-Based Reasoning Research and Development*, pages 378–392. Springer, 2011. 20

[45] Maja Pantic and Leon Rothkrantz. **Case-Based Reasoning for User-Profiled Recognition of Emotions from Face Images**. In *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on Multimedia and Expo*, **1**, pages 391–394. IEEE, 2004. 20

[46] George A Miller. **The Cognitive Revolution: A Historical Perspective**. *Trends in Cognitive Sciences*, **7**(3):141–144, 2003. 21

[47] Michael M Richter and Stefan Wess. *Similarity, Uncertainty and Case-Based Reasoning in PATDEX*. Springer, 1991. 21

[48] **Uncertainty**. `http://www.wi2.uni-trier.de/eccbr08/index.php-task=workshops.htm`. Accessed: 2016-03-24. 21

[49] Jerzy Surma. **Case-Based Approach for Supporting Strategy Decision Making**. *Expert Systems*, **32**(4):546–554, 2015. 21

[50] Eyke Hüllermeier. *Computational Intelligence: Theory and Applications International Conference, 6th Fuzzy Days Dortmund, Germany, May 25–28 1999 Proceedings*, chapter A Possibilistic Formalization of Case-Based Reasoning and Decision Making, pages 411–420. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. 21

[51] Belén Díaz-Agudo, Pablo Gervás, and Federico Peinado. **A Case-Based Reasoning Approach to Story Plot Generation**. In *Advances in Case-Based Reasoning*, pages 142–156. Springer, 2004. 22

[52] Mary Lou Maher, Muthaukumar Balachandran, and Dong Mei Zhang. *Case-Based Reasoning in Design*. Psychology Press, 1995. 22

[53] Janet L Kolodner. **Improving Human Decision Making through Case-Based Decision Aiding**. *AI Magazine*, **12**(2):52, 1991. 22

[54] S Dutta, B Wierenga, and A Dalebout. **Case-Based Reasoning Systems: from Automation to Decision-Aiding and Stimulation**. *Knowledge and Data Engineering, IEEE Transactions on Knowledge and Data Engineering*, **9**(6):911–922, Nov 1997. 22, 28, 29

[55] Thomas R Roth-Berghofer. *Knowledge Maintenance of Case-Based Reasoning Systems: the SIAM Methodology*, **262**. IOS Press, 2003. 22

[56] Janet L Kolodner, Michael T Cox, and Pedro A González-Calero. **Case-Based Reasoning-Inspired Approaches to Education**. *The Knowledge Engineering Review*, **20**(03):299–303, 2005. 22

[57] Kristian J Hammond, Robin D Burke, and Steven L Lytinen. **A Case-Based Approach to Knowledge Navigation.** In *IJCAI*, pages 2071–2072, 1995. 22

[58] Isabelle Bichindaritz, Emin Kansu, and Keith M Sullivan. **Case-Based Reasoning in Care-Partner: Gathering Evidence for Evidence-Based Medical Practice**. In *Advances in Case-Based Reasoning*, pages 334–345. Springer, 1998. 22

[59] Kristian J Hammond. **Explaining and Repairing Plans that Fail**. *Artificial Intelligence*, **45**(1):173–228, 1990. 23

[60] Robin Burke. **The Wasabi Personal Shopper: A Case-Based Recommender System**. In *AAAI/IAAI*, pages 844–849, 1999. 26

[61] Lorraine Mcginty and Barry Smyth. **Adaptive Selection: An Analysis of Critiquing and Preference-Based Feedback in Conversational Recommender Systems**. *International Journal of Electronic Commerce*, **11**(2):35–57, 2006. 27

[62] Lorraine Mc Ginty and Barry Smyth. **Comparison-Based Recommendation**. In *Advances in Case-Based Reasoning*, pages 575–589. Springer, 2002. 27

[63] Hui Li, Hojjat Adeli, Jie Sun, and Jian-Guang Han. **Hybridizing Principles of TOPSIS with Case-Based Reasoning for Business Failure Prediction**. *Computers & Operations Research*, **38**(2):409–419, 2011. 27

[64] David R Kraay and Patrick T Harker. **Case-Based Reasoning for Repetitive Combinatorial Optimization Problems, Part I: Framework**. *Journal of Heuristics*, **2**(1):55–85, 1996. 34, 108

[65] Stephan Grolimund and Jean-Gabriel Ganascia. **Driving Tabu Search with Case-Based Reasoning**. *European Journal of Operational Research*, **103**(2):326 – 338, 1997. 34

[66] Eyke Hüllermeier. **Focusing Search by Using Problem Solving Experience**. In *ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, Germany, August 20-25, 2000*, pages 50–54, 2000. 34

[67] Steven Robertson. **An Introduction to Decision Theory**. *The Bulletin of Symbolic Logic*, **16**(3):pp. 413–415, 2010. 39

[68] Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. **Distance Metric Learning with Application to Clustering with Side-Information**. *Advances in Neural Information Processing Systems*, pages 521–528, 2003. 44

[69] Mark E Glickman and Shane T Jensen. **Adaptive Paired Comparison Design**. *Journal of Statistical Planning and Inference*, **127**:2005, 2005. 44, 46

[70] Armin Stahl. **Learning Similarity Measures: A Formal View Based on a Generalized CBR Model**. In Hctor Muoz-vila and Francesco Ricci, editors, *Case-Based Reasoning Research and Development*, **3620** of *Lecture Notes in Computer Science*, pages 507–521. Springer Berlin Heidelberg, 2005. 45, 47, 56

[71] Ning Xiong and Peter Funk. **Combined Feature Selection and Similarity Modelling in Case-Based Reasoning Using Hierarchical Memetic Algorithm**. In *2010 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–6. IEEE, 2010. 45

[72] Maria Salamó and Elisabet Golobardes. **Analysing Rough Sets Weighting Methods for Case-Based Reasoning Systems.** *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, **6**(15):0, 2002. 45

[73] Wolfgang Wilke and Ralph Bergmann. **Considering Decision Cost During Learning of Feature Weights**. In *Advances in Case-Based Reasoning*, pages 460–472. Springer, 1996. 45

[74] Aurélien Bellet, Amaury Habrard, and Marc Sebban. **A Survey on Metric Learning for Feature Vectors and Structured Data**. *CoRR*, **abs/1306.6709**, 2013. 46

[75] Roger R Davidson and Daniel L Solomon. **A Bayesian Approach to Paired Comparison Experimentation**. *Biometrika*, **60**(3):pp. 477–487, 1973. 46

[76] Aaron Wilson, Alan Fern, and Prasad Tadepalli. **A Bayesian Approach for Policy Learning from Trajectory Preference Queries**. In P. Bartlett, F.c.n. Pereira, C.j.c. Burges, L. Bottou, and K.q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1142–1150. NIPS, 2012. 46

[77] Liu Yang, Rong Jin, and Rahul Sukthankar. **Bayesian Active Distance Metric Learning**. *CoRR*, **abs/1206.5283**, 2012. 46

[78] Shengbo Guo and Scott Sanner. **Real-Time Multiattribute Bayesian Preference Elicitation with Pairwise Comparison Queries**. In *International Conference on Artificial Intelligence and Statistics*, pages 289–296, 2010. 46

[79] Can Burak. **Weighted Distances Between Preferences**. Technical report, Maastricht University, Maastricht Research School of Economics of Technology and Organization (METEOR), 2012. 46

[80] Vu Ha and Peter Haddawy. **Toward Case-Based Preference Elicitation: Similarity Measures on Preference Structures**. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 193–201. Morgan Kaufmann Publishers Inc., 1998. 46

[81] Nan Li, William Cushing, Subbarao Kambhampati, and Sung Wook Yoon. **Learning User Plan Preferences Obfuscated by Feasibility Constraints.** In *ICAPS*, 2009. 47

[82] Armin Stahl. **Learning Feature Weights From Case Order Feedback**. In *Proceedings of the 4th International Conference on Case-based Reasoning*, pages 502–516. Springer, 2001. 47, 55

[83] Armin Stahl and Sascha Schmitt. **Optimizing Retrieval in CBR by Introducing Solution Similarity**. In *Proceedings of the International Conference on Artificial Intelligence (IC-AI'02)*. CSREA Press, 2002. 47

[84] Armin Stahl. **Optimizing Similarity Assessment in Case-Based Reasoning**. In *21st National Conference on Artificial Intelligence (AAAI-06)*. AAAI Press, 2006. 47

[85] Armin Stahl and Thomas Gabel. **Using Evolution Programs to Learn Local Similarity Measures**. In KevinD. Ashley and DerekG. Bridge, editors, *Case-Based Reasoning Research and Development*, **2689** of *Lecture Notes in Computer Science*, pages 537–551. Springer Berlin Heidelberg, 2003. 47, 55

[86] Zhong Zhang and Qiang Yang. **Dynamic Refinement of Feature Weights Using Quantitative Introspective Learning**. In *IJCAI*, pages 228–233, 1999. 47

[87] Hans-Dieter Burkhard and Michael M Richter. **On the Notion of Similarity in Case-Based Reasoning and Fuzzy Theory**. In *Soft Computing in Case-Based Reasoning*, pages 29–45. Springer, 2001. 50

[88] Thomas Martinetz, Kai Labusch, and Daniel Schneegass. **Softdoublemaxminover: perceptron-like training of support vector machines**. *Neural Networks, IEEE Transactions on*, **20**(7):1061–1072, 2009. 55, 58

[89] Andrea Bonzano, Pdraig Cunningham, and Barry Smyth. **Using Introspective Learning to Improve Retrieval in CBR: A Case Study in Air Traffic Control**. In DavidB. Leake and Enric Plaza, editors, *Case-Based Reasoning Research and Development*, **1266** of *Lecture Notes in Computer Science*, pages 291–302. Springer Berlin Heidelberg, 1997. 55

[90] Igor Kononenko. **Estimating Attributes: Analysis and Extensions of RELIEF**. In *Springer Verlag*, pages 171–182. Springer Verlag, 1994. 55

[91] Francesco Ricci and Paolo Avesani. **Learning a Local Similarity Metric for Case-Based Reasoning**. In *International Conference on Case-Based Reasoning (ICCBR-95)*, pages 301–312. Springer-Verlag, 1995. 55

[92] Dietrich Wettschereck and David W Aha. **Weighting Features**. In Manuela Veloso and Agnar Aamodt, editors, *Case-Based Reasoning Research and Development*, **1010** of *Lecture Notes in Computer Science*, pages 347–358. Springer Berlin Heidelberg, 1995. 56

[93] Yingquan Wu, Krasimir G Ianakiev, and Venu Govindaraju. **Improvements in K-Nearest Neighbor Classification**. In Sameer Singh, Nabeel Murshed, and Walter Kropatsch, editors, *Advances in Pattern Recognition ICAPR 2001*, **2013** of *Lecture Notes in Computer Science*, pages 224–231. Springer Berlin Heidelberg, 2001. 56

[94] Godfried Toussaint. **Geometric Proximity Graphs for Improving Nearest Neighbor Methods in Instance-Based Learning and Data Mining**. *International Journal of Computational Geometry & Applications*, **15**(02):101–150, 2005. 56

[95] Thomas Roth-Berghofer and Ioannis Iglezakis. **Six Steps in Case-Based Reasoning: Towards a Maintenance Methodology for Case-Based Reasoning Systems**. In *Professionelles Wissensmanagement: Erfahrungen und Visionen (includes the Proceedings of the 9th German Workshop on Case-Based Reasoning GWCBR)*, pages 198–208. Shaker-Verlag, 2001. 64

[96] Barry Smyth and Mark T. Keane. **Remembering to Forget: A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems**. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, pages 377–382, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. 64, 65

[97] Jim Zhu and Qiang Yang. **Remembering to Add: Competence-Preserving Case-Addition Policies for Case-Base Maintenance**. In *Proceedings IJCAI–99, 16th International Joint Conference on Artificial Intelligence*, pages 234–239, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. 64, 65

[98] Barry Smyth. **Case-Base Maintenance**. In Angel Pasqual del Pobil, Jos Mira, and Moonis Ali, editors, *Tasks and Methods in Applied Artificial*

*Intelligence*, **1416** of *Lecture Notes in Computer Science*, pages 507–516. Springer Berlin Heidelberg, 1998. 64

[99] A. Lawanna and J. Daengdej. **Hybrid Technique and Competence-Preserving Case Deletion Methods for Case Maintenance in Case-Based Reasoning**. International Journal of Engineering Science and Technology, 2010. 64

[100] Maria Salamo , Elisabet Golobardes, Enginyeria Arquitectura, and La Salle. **Hybrid Deletion Policies for Case Base Maintenance**. In *Proceedings of FLAIRS-2003*, pages 150–154, 2003. 65

[101] Geoffrey W. Gates. **The Reduced Nearest Neighbor Rule**. *IEEE Transactions on Information Theory*, **18**(3):431–433, 1972. 65

[102] P Hart. **The Condensed Nearest Neighbor Rule**. *IEEE Transactions on Information Theory*, **14**(3):515–516, May 1968. 65

[103] Maria Salamo and Elisabet Golobardes. **Rough Sets Reduction Techniques for Case-Based Reasoning**. In David W. Aha and Ian Watson, editors, *Case-Based Reasoning Research and Development*, pages 467–482. Springer Berlin Heidelberg, 2001. 65

[104] Lisa Cummins and Derek Bridge. **On Dataset Complexity for Case Base Maintenance**. In *Proc. ICCBR–2011, 19th International Conference on Case-Based Reasoning*, pages 47–61, London, UK, 2011. Springer-Verlag. 65

[105] Santiago Ontanon and Enric Plaza. **Justification-Based Selection of Training Examples for Case Base Reduction**. In F. Esposito F., Giannotti and D. Pedresh, editors, *Proceedings ECML–2004, European Conference on Machine Learning*, number 3201(15) in Lecture Notes in Artificial Intelligence, pages 310–321. Springer-Verlag, 2004. 65

[106] Susan Craw, Stewart Massie, and Nirmalie Wiratunga. **Informed Case Base Maintenance: A Complexity Profiling Approach**. In *Proceedings AAAI–2007, Twenty-Second National Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1618–1621, 2007. 65

[107] Eduardo Lupiani, Jose M Juarez, and Jose Palma. **Evaluating Case Base Maintenance Algorithms**. *Knowledge-Based Systems*, **67**:180 – 194, 2014. 66

[108] Xin-She Yang. *Nature-Inspired Optimization Algorithms*. Elsevier, 2014. 78

[109] Theerayod Wiangtong, Peter YK Cheung, and Wayne Luk. **Comparing Three Heuristic Search Methods for Functional Partitioning in Hardware–Software Codesign**. *Design Automation for Embedded Systems*, **6**(4):425–449, 2002. 79, 82, 84

[110] Fred Glover and Harvey J Greenberg. **New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intelligence**. *European Journal of Operational Research*, **39**(2):119 – 130, 1989. 79, 82, 84

[111] Richard E Korf. **Linear-Space Best-First Search**. *Artificial Intelligence*, **62**(1):41–78, 1993. 79

[112] Stuart Russell and Peter Norvig. **A Modern Approach**. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, **25**:27, 1995. 79, 80, 81, 82, 86

[113] **A star algorithm**. `http://intelligence.worldofcomputing.net/ai-search/a-star-algorithm.html`. Accessed: 2016-03-30. 80, 81

[114] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. *Iterated Local Search*. Springer, 2003. 81

[115] Bart Selman and Carla P Gomes. **Hill-Climbing Search**. *Encyclopedia of Cognitive Science*, 2006. 81, 82

[116] Michel Gendreau and Jean-Yves Potvin. **Tabu Search**. In EdmundK. Burke and Graham Kendall, editors, *Search Methodologies*, pages 165–186. Springer US, 2005. 82

[117] Edward J Anderson and Michael C Ferris. **A Direct Search Algorithm for Optimization with Noisy Function Evaluations**. *SIAM Journal on Optimization*, **11**(3):837–857, 2001. 83

[118] David E Goldberg and John H Holland. **Genetic Algorithms and Machine Learning**. *Machine Learning*, **3**(2):95–99, 1988. 84

[119] William L Goffe, Gary D Ferrier, and John Rogers. **Global Optimization of Statistical Functions with Simulated Annealing**. *Journal of Econometrics*, **60**(1):65–99, 1994. 84

[120] James Kennedy. **Particle Swarm Optimization**. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2011. 85

[121] Riccardo Poli, James Kennedy, and Tim Blackwell. **Particle Swarm Optimization**. *Swarm Intelligence*, **1**(1):33–57, 2007. 85

[122] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. **Practical Bayesian Optimization of Machine Learning Algorithms**. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012. 86

[123] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando De Freitas. **Bayesian Optimization in High Dimensions via Random Embeddings.** In *IJCAI*. Citeseer, 2013. 86

[124] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. **Bayesian Optimization for Learning Gaits Under Uncertainty**. *Annals of Mathematics and Artificial Intelligence*, pages 1–19, 2015. 87

[125] Pedro Domingos. **A Few Useful Things to Know About Machine Learning**. *Commun. ACM*, **55**(10):78–87, October 2012. 88

[126] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. **Large Margin Methods for Structured and Interdependent Output Variables**. *J. Mach. Learn. Res.*, **6**:1453–1484, December 2005. 89

[127] Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. **Structured Prediction via Output Space Search**. *Journal of Machine Learning Research*, **15**:1317–1350, 2014. 89

[128] J Rao Doppa, A Fern, and P Tadepalli. **Output Space Search for Structured Prediction**. *ArXiv e-prints*, June 2012. 89

[129] S Hare, A Saffari, and P H S Torr. **Struck: Structured Output Tracking with Kernels**. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 263–270, Nov 2011. 90

[130] David Pierce, , David Pierce, and Claire Cardie. **User-Oriented Machine Learning Strategies for Information Extraction: Putting the Human Back in the Loop**. In *In the loop, In Working*, 2001. 90

[131] Riad Akrour, Marc Schoenauer, Michèle Sebag, and Jean-Christophe Souplet. **Programming by Feedback**. In *International Conference on Machine Learning*, number 32 in JMLR Proceedings, pages 1503–1511, Pékin, China, June 2014. JMLR.org. 91

[132] Andrea L Thomaz and Cynthia Breazeal. **Reinforcement Learning with Human Teachers: Evidence of Feedback and Guidance with Implications for Learning Performance**. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 1000–1005. AAAI Press, 2006. 91

[133] Jerry Alan Fails and Dan R. Olsen, Jr. **Interactive Machine Learning**. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, pages 39–45, New York, NY, USA, 2003. ACM. 91, 92

[134] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. **Visual Recognition with Humans in the Loop**. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision ECCV 2010*, **6314** of *Lecture Notes in Computer Science*, pages 438–451. Springer Berlin Heidelberg, 2010. 91, 92

[135] A Waibel, R Stiefelhagen, R Carlson, J Casas, J Kleindienst, L Lamel, O Lanz, D Mostefa, M Omologo, F Pianesi, L Polymenakos, G Potamianos, J Soldatos, G Sutschet, and J Terken. *Computers in the Human Interaction Loop*, page 1071–1116. Springer, Boston, MA, 2010. 91

[136] Wenchao Li, Dorsa Sadigh, S Shankar Sastry, and SanjitA Seshia. **Synthesis for Human-in-the-Loop Control Systems**. In Erika brahm and Klaus Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, **8413** of *Lecture Notes in Computer Science*, pages 470–484. Springer Berlin Heidelberg, 2014. 91

[137] Aaron Steinfeld, S Rachael Bennett, Kyle Cunningham, Matt Lahut, Pablo-Alejandro Quinones, Django Wexler, Dan Siewiorek, Jordan Hayes, Paul Cohen, Julie Fitzgerald, et al. **Evaluation of an Integrated Multi-Task Machine Learning System with Humans in the Loop**. In *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems*, pages 168–174. ACM, 2007. 92

[138] C Brodley, A Kak, C Shyu, J Dy, L Broderick, and A M Aisen. **Content-Based Retrieval from Medical Image Database: A Synergy of Human Interaction, Machine Learning, and Computer Vision**. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 760–767, 1999. 92

[139] Roberto Battiti, Mauro Brunato, and Franco Mascia. *Reactive Search and Intelligent Optimization*, **45** of *Operations Research/Computer Science Interfaces*. Springer Verlag, 2008. 93, 94, 95

[140] Roberto Battiti. **Reactive Search: Toward Self-Tuning Heuristics**. *Modern Heuristic Search Methods*, pages 61–83, 1996. 93

[141] Roberto Battiti and Giampietro Tecchiolli. **The Reactive Tabu Search**. *ORSA Journal on Computing*, **6**(2):126–140, 1994. 93

[142] Roberto Battiti and Mauro Brunato. *The LION way. Machine Learning* plus *Intelligent Optimization*. LIONlab, University of Trento, Italy, 2014. 94

[143] Zhipeng L and Jin-Kao Hao. **A Critical Element-Guided Perturbation Strategy for Iterated Local Search**. In Carlos Cotta and Peter Cowling, editors, *Evolutionary Computation in Combinatorial Optimization*, **5482** of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2009. 94

[144] Everardo Gutirrez and CarlosA. Brizuela. **ILS-Perturbation Based on Local Optima Structure for the QAP Problem**. In Alexander Gelbukh and CarlosAlberto Reyes-Garcia, editors, *MICAI 2006: Advances in Artificial Intelligence*, **4293** of *Lecture Notes in Computer Science*, pages 404–414. Springer Berlin Heidelberg, 2006. 94

[145] Mark A Hall and Lloyd A Smith. **Feature Selection for Machine Learning: Comparing a Correlation-Based Filter Approach to the Wrapper**. In *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, pages 235–239. AAAI Press, 1999. 95

[146] Mazen W Karaman, Sanna Herrgard, Daniel K Treiber, Paul Gallant, Corey E Atteridge, Brian T Campbell, Katrina W Chan, Pietro Ciceri, Mindy I Davis, Philip T Edeen, et al. **A Quantitative Analysis of Kinase Inhibitor Selectivity**. *Nature Biotechnology*, **26**(1):127–132, 2008. 100

[147] Stefan Schmitt, Daniel Kuhn, and Gerhard Klebe. **A New Method to Detect Related Function Among Proteins Independent of Sequence and Fold Homology**. *Journal of Molecular Biology*, **323**(2):387–406, 2002. 100

[148] Michiel Stock. *Learning Pairwise Relations in Bioinformatics: Three Case Studies*. PhD thesis, Masters Thesis, University of Ghent, 2012. 101

[149] Zoubin Ghahramani and Katherine A Heller. **Bayesian Sets**. In *NIPS*, **2**, pages 22–23, 2005. 102

[150] Arthur Asuncion and David Newman. **UCI Machine Learning Repository**, 2007. 104

[151] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. **Modeling Wine Preferences by Data Mining from Physicochemical Properties**. *Decision Support Systems*, **47**(4):547–553, 2009. 104

[152] Pádraig Cunningham, Barry Smyth, and Neil Hurley. *On the Use of CBR in Optimisation Problems such as the TSP*. Springer Berlin Heidelberg, 1995. 108

[153] Hossein Erfani. **Integrating Case-Based Reasoning, Knowledge-Based Approach and TSP Algorithm for Minimum Tour Finding**. *Journal of Operational Research and Its Applications (Journal of Applied Mathematics)*, 2006. 108

[154] Mustafa Akgül. *The Linear Assignment Problem*. Springer, 1992. 109

[155] Alex Krizhevsky and Geoffrey Hinton. **Learning Multiple Layers of Features from Tiny Images**, 2009. 113

[156] Alex Zvoleff. **Package glcm**. 2015. 115

[157] **Percentage difference between images**. `https://rosettacode.org/wiki/Percentage_difference_between_images`. Accessed: 2016-03-23. 115

## Publications


- Abdel-Aziz, A., Cheng, W., Strickert, M., Hüllermeier, E. (2013). Preference-based CBR: A Search-Based Problem Solving Framework. In 21st International Conference in Case-Based Reasoning , ICCBR 2013, Saratoga Springs, NY, USA, pp. 1-14, July 8-11, 2013.

- Abdel-Aziz, A., Strickert, M., Fober, T., Hüllermeier, E. Protein Structure Retrieval Using Preference-Based CBR. ICCBR 2013 - Workshop, Saratoga Springs, NY, USA, 2013.

- Abdel-Aziz, A., Strickert, M., Hüllermeier, E. (2014). Learning Solution Similarity in Preference-Based CBR. In 22nd International Conference in Case-Based Reasoning , ICCBR 2014, Cork, Ireland, pp. 17-31, September 29, 2014 - October 1, 2014.

- Abdel-Aziz, A., Hüllermeier, E. (2015). Case Base Maintenance in Preference-Based CBR. In 23rd International Conference in Case-Based Reasoning, ICCBR 2015, Frankfurt am Main, Germany, pp. 1-14, September 28-30, 2015.

# Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This document has not previously been presented in identical or similar form to any other German or foreign examination board.

The thesis work was conducted from July 2012 to June 2016 under the supervision of Prof. Eyke Hüllermeier at the University of Marburg.

Marburg,