Preference-based CBR: A Search-based Problem Solving Framework

Amira Abdel-Aziz, Weiwei Cheng, Marc Strickert, Eyke Hüllermeier

Department of Mathematics and Computer Science
Marburg University, Germany
{amira,cheng,strickert,eyke}@mathematik.uni-marburg.de

Abstract. 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. This paper is a continuation of recent work on a formalization of preference-based CBR that was focused on an essential part of the methodology: a method to predict a most plausible candidate solution given a set of preferences on other solutions, deemed relevant for the problem at hand. Here, we go one step further by embedding this method in a more general search-based problem solving framework. 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. The effectiveness of this approach is illustrated in two case studies, one from the field of bioinformatics and the other one related to the computer cooking domain.

1 Introduction

A preference-based approach to case-based reasoning (CBR) has recently been advocated in [1]. Building on general ideas and concepts for preference handling in artificial intelligence (AI), which have already been applied successfully in other fields [2–4], 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.

In fact, as argued in [1], a preference-based approach to CBR appears to be appealing for several reasons, notably because case-based experiences lend themselves to representations in terms of preference relations quite naturally. Moreover, 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 (constraint-based) one, which have already been observed for AI in general, seem to apply to CBR in particular. These advantages are nicely explained in [5]: "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."

In [1], we made a first step toward preference-based CBR 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. In this paper, we go one step further by embedding this inference procedure in a more general, search-based problem solving framework. 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.

The remainder of the paper is organized as follows. By way of background, Section 2 recapitulates the main ideas of Pref-CBR, and Section 3 briefly recalls the case-based inference procedure of [1]. Although these two sections are to some extent redundant, they are included here to increase readability of the paper and to make it more self-contained. In Section 4, we introduce and detail our search-based problem solving framework. In Section 5, two case studies are presented to illustrate the effectiveness of this approach, one from the field of bioinformatics (molecular docking, drug discovery) and the other one related to the computer cooking domain. The paper ends with some concluding remarks and an outlook on future work in Section 5.

2 Preference-based CBR

2.1 Conventional 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.

- *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.

Table 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, Δ_X	similarity/distance measure on $\mathbb{X}$
S_Y, Δ_Y	similarity/distance measure on $\mathbb{Y}$
$\mathcal{N}(\boldsymbol{y})$	neighborhood of a solution $\boldsymbol{y}$
$\mathfrak{P}(\mathbb{Y})$	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 (4))

2.2 Preference-based Knowledge Representation

To avoid these problems, preference-based CBR replaces experiences of the form "solution $\boldsymbol{y}$ (optimally) solves problem $\boldsymbol{x}$" by information of the form "$\boldsymbol{y}$ is better (more preferred) than $\boldsymbol{z}$ as a solution for $\boldsymbol{x}$". 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 obviously overcomes the problems discussed above. As soon as two candidate solutions $\boldsymbol{y}$ and $\boldsymbol{z}$ have been tried as solutions for a problem $\boldsymbol{x}$, 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 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:

$$\boldsymbol{y}_1 \succeq_{\boldsymbol{x}} \boldsymbol{y}_2 \succeq_{\boldsymbol{x}} \boldsymbol{y}_3 \succeq_{\boldsymbol{x}} \cdots \succeq_{\boldsymbol{x}} \boldsymbol{y}_n \tag{1}$$

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

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?

2.3 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 that the solution space $\mathbb{Y}$ to be equipped with a similarity measure S_Y or, equivalently, with a (reciprocal) distance measure Δ_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 Δ_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 Δ_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.*

3 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 $\succeq_{\boldsymbol{x}}$ or, equivalently, what is the most likely "ideal" solution $\boldsymbol{y}^*$ for $\boldsymbol{x}_0$?

3.1 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 θ 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 θ via maximum likelihood (ML) by

$$\theta^{ML} = \arg\max_{\theta \in \Theta} \mathbf{P}_\theta(\mathcal{D}) \ . \tag{2}$$

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.

3.2 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 [6]. 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 [1], more specific assumptions on an underlying (latent) utility function on solutions justify the *logit* model of discrete choice:

$$\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z}) \;=\; \Big(1 + \exp\big(-\beta(\Delta_Y(\boldsymbol{z}, \boldsymbol{y}^*) - \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*))\big)\Big)^{-1} \tag{3}$$

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 β can be seen as a measure of precision of the preference feedback. For large β, the probability (3) converges toward 0 if $\Delta_Y(\boldsymbol{z}, \boldsymbol{y}^*) < \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)$ and toward 1 if $\Delta_Y(\boldsymbol{y}', \boldsymbol{y}^*) > \Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*)$; 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.

3.3 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

$$\ell(\theta) = \ell(\boldsymbol{y}, \beta) = -\sum_{i=1}^{N} \log\Big(1 + \exp\big(-\beta(\Delta(\boldsymbol{z}^{(i)}, \boldsymbol{y}) - \Delta(\boldsymbol{y}^{(i)}, \boldsymbol{y}))\big)\Big). \tag{4}$$

The maximum likelihood estimation (MLE) $\theta_{ML} = (\boldsymbol{y}^{ML}, \beta^{ML})$ of θ^* is given by the maximizer of (4):

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

The problem of finding this estimation in an efficient way is discussed in [1].

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 [7–9]. The other way around, the CBR process itself can be formalized as a search process, namely a traversal of the space of potential solutions [10]. 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) which, 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 Δ_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 by an upper bound L reflects our assumption that an evaluation of a candidate solution is costly.
- 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 (referred to as CBI in the pseudo-code) described in Section 3 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

$$\Delta\left(\boldsymbol{y}^*, \boldsymbol{y} \succ \boldsymbol{z}\right) = \min\left\{\Delta_Y\left(\boldsymbol{y}^*, \boldsymbol{y}\right), \Delta_Y\left(\boldsymbol{y}^*, \boldsymbol{z}\right)\right\} \quad , \tag{5}$$

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

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

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 Δ_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

smaller J, the less preferences are used, i.e., the more local the determination of the direction of the search process[2] 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 (5) 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 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 $\boldsymbol{x}_0$.

Although we did not implement this alternative so far, let us mention that a stochastic component can be added to our search procedure in a quite natural way. 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).

5 Case Studies

5.1 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.

[2] 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.

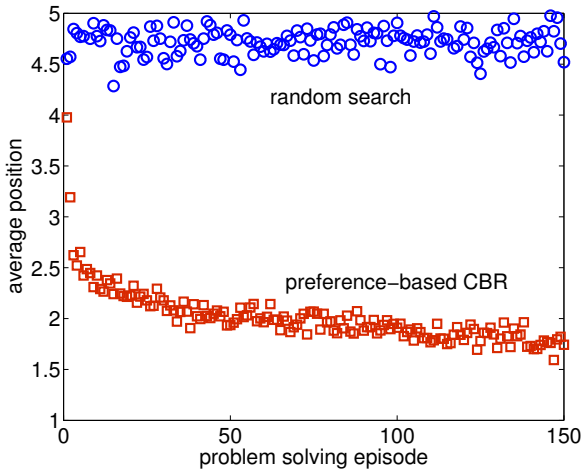

Fig. 1. Average performance of Pref-CBR and random search on the drug discovery problem in the first 150 problem solving episodes.

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 [11]. 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 [12]. For the solutions (ligands), a similarity S_Y was determined based on molecular fingerprints derived from the SMILES code using a molecular operating environment. These fingerprints were used to create a graph representation of the molecules, for which the Tanimoto similarity was determined [13]. Both similarities S_X and S_Y were normalized to the unit interval, and corresponding distances Δ_X and Δ_Y were defined as $1 - S_X$ and $1 - S_Y$, respectively.

We applied Algorithm 1 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 $\mathcal{N}(\boldsymbol{y}) = \mathbb{Y} \setminus \{\boldsymbol{y}\}$. 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. Fig. 1 shows the results for parameters $K = 3$, $L = 5$, $J = 15$ in Algorithm 1 (other settings let 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.

5.2 The Set Completion Problem

In a second experiment, we considered a set completion problem that is similar to the problem solved by the Bayesian set algorithm proposed in [14]. 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 Δ_X and Δ_Y in terms of the size of the symmetric difference Δ, 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 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:

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

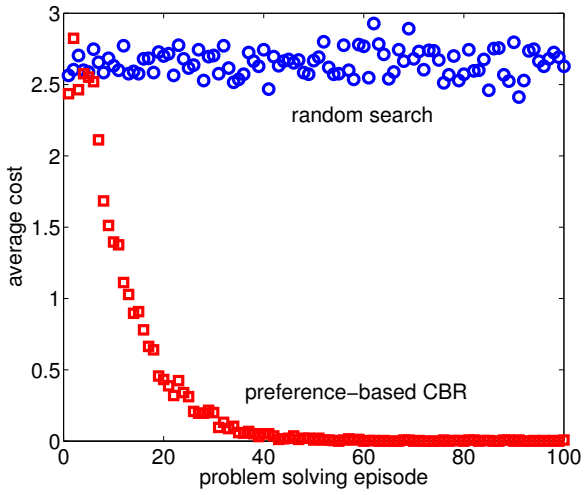

Fig. 2. Average performance of Pref-CBR and random search on the set completion problem in the first 100 problem solving episodes.

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

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 Fig. 2, which are qualitatively similar to those of the previous study.

6 Conclusion

In this paper, 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 has been illustrated in two concrete case studies.

For future work, we plan to extend and generalize our framework in various directions. First, the search procedure presented here can essentially be seen as a preference-based variant of a simple hill-climbing method. Needless to say,

the idea of using preferences for guiding the search process can be applied to other, more sophisticated search methods (including population-based stochastic search algorithms) in a quite similar way. Second, since the number of preferences collected in the course of time may become rather large, effective methods for case base maintenance ought to be developed. Third, 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 2.3). Therefore, a proper specification of this measure is a prerequisite for the effectiveness of our preference-guided search procedure. It would hence 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. The method for learning similarity measures from qualitative feedback proposed in [15] appears to be ideally suited for this purpose.

Acknowledgments

This work has been supported by the German Research Foundation (DFG). We are grateful to Peter Kolb and Denis Schmidt for providing us the data used for in the drug discovery experiment.

References

1. E. Hüllermeier and P. Schlegel. Preference-based CBR: First steps toward a methodological framework. In A. Ram and N. Wiratunga, editors, *Proceedings ICCBR–2011, 19th International Conference on Case-Based Reasoning*, pages 77–91. Springer-Verlag, 2011.
2. J. Doyle. Prospects for preferences. *Comput. Intell.*, 20(2):111–136, 2004.
3. J. Goldsmith and U. Junker. Special issue on preference handling for Artificial Intelligence. *Computational Intelligence*, 29(4), 2008.
4. C. Domshlak, E. Hüllermeier, S. Kaci, and H. Prade. Preferences in AI: An overview. *Artificial Intelligence*, 2011.
5. R.I. Brafman and C. Domshlak. Preference handling–an introductory tutorial. *AI Magazine*, 30(1), 2009.
6. M. Peterson. *An Introduction to Decision Theory.* Cambridge Univ. Press, 2009.
7. D.R. Kraay and P.T. Harker. Case-based reasoning for repetitive combinatorial optimization problems, part I: Framework. *Journal of Heuristics*, 2:55–85, 1996.
8. S. Grolimund and J.G. Ganascia. Driving tabu search with case-based reasoning. *European Journal of Operational Research*, 103(2):326–338, 1997.
9. E. Hüllermeier. Focusing search by using problem solving experience. In W. Horn, editor, *Proceedings ECAI–2000, 14th European Conference on Artificial Intelligence*, pages 55–59, Berlin, Germany, 2000. IOS Press.
10. R. Bergmann and W. Wilke. Towards a new formal model of transformational adaptation in case-based reasoning. In H. Prade, editor, ECAI-98*, 13th European Conference on Artificial Intelligence*, pages 53–57, 1998.
11. M.W. Karaman et al. A quantitative analysis of kinase inhibitor selectivity. *Nature Biotechnology*, 26:127–132, 2008.

12. S. Schmitt, D. Kuhn, and G. 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.
13. M. Stock. Learning pairwise relations in bioinformatics: Three case studies. Master's thesis, University of Ghent, 2012.
14. Z. Ghahramani and K.A. Heller. Bayesian sets. In *Proceedings NIPS–2005*, 2005.
15. W. Cheng and E. Hüllermeier. Learning similarity functions from qualitative feedback. In K.D. Althoff, R. Bergmann, M. Minor, and A. Hanft, editors, *Proceedings ECCBR–2008, 9th European Conference on Case-Based Reasoning*, number 5239 in LNAI, pages 120–134, Trier, Germany, 2008. Springer-Verlag.