Case Base Maintenance in Preference-based CBR

Amira Abdel-Aziz[1] and Eyke Hüllermeier[2]

[1] Department of Mathematics and Computer Science
University of Marburg, Germany
amira@mathematik.uni-marburg.de
[2] Department of Computer Science
University of Paderborn, Germany
eyke@upb.de

Abstract. In preference-based CBR (Pref-CBR), 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. Since a potentially large number of such preferences can be collected in the course of each problem solving episode, case base maintenance clearly becomes an issue in Pref-CBR. In this paper, we therefore extend our Pref-CBR framework by another component, namely, a method for dynamic case base maintenance. The main goal of this method is to increase efficiency of case-based problem solving, by reducing the size of the case base, while maintaining performance. To illustrate the effectiveness of our approach, we present a case study in which Pref-CBR is used for the repetitive traveling salesman problem.

1 Introduction

In the recent years, we have been working toward a methodological framework for case-based reasoning on the basis of formal concepts and methods for reasoning with *preferences* [10, 1, 2]. Deviating from the common representation of experiences in terms of problem/solution tuples $(\boldsymbol{x}, \boldsymbol{y})$, preference-based CBR (or Pref-CBR for short) proceeds from weaker "chunks of information" $\boldsymbol{y} \succ_{\boldsymbol{x}} \boldsymbol{z}$, namely, preferences between competing solutions "contextualized" by problems: $\boldsymbol{y}$ is (likely to be) more preferred than $\boldsymbol{z}$ as a solution for $\boldsymbol{x}$.

Problem solving in Pref-CBR is realized as a search process, in which candidate solutions are iteratively improved. 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).

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* [18, 11]. 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, it might be even more critical for preference-based than for conventional CBR.

In this paper, we therefore address the problem of case base maintenance in Pref-CBR. To this end, we develop a CBM strategy that extends our Pref-CBR framework so far. 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 remainder of the paper is organized as follows. Related work on case base maintenance is briefly recalled in Section 2. By way of background, and to assure a certain level of self-containedness, we also recall the essentials of Pref-CBR in Section 3. Our approach to case base maintenance for Pref-CBR is then detailed in Section 4. To illustrate the effectiveness of this approach, Section 5 presents a case study using the (repetitive) traveling salesman problem as a problem solving domain. The paper ends with some concluding remarks and an outlook on future work in Section 6.

2 Related Work

To increase efficiency while maintaining the competence of a case base, several CBR methods implement strategies that focus on choosing which 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 [21]; 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); cases with negative utility are removed [20]. There are other methods such as footprint deletion and footprint utility deletion, which specify the cases to be deleted based on their competence contributions [19]. The cases are categorized into pivotal, spanning, support and auxiliary; pivotal cases have highest effect on competence, while auxiliary cases have lowest effect [13]. Modifying the idea of coverage (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 [19], cases are identified by their coverage and reachability values based on rough set theory for categorizing data in [17], 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 [8]. This in-

crease in efficiency could in return cause some degradation in performance. One well-known method is based on the condensed nearest neighbor (CNN) rule by [9], 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 [16]. 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 [5] though. Also described in [5], 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 [15], 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 [21], but in the former the selection of cases is based on the justifications rather than the competence.

Additionally, 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 [11]. Complexity-informed maintenance is another method presented in [4]; 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 [14], 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 method, which is specifically designed for the Pref-CBR problem solving framework.

3 Preference-based CBR in a Nutshell

In this section, we briefly recall the basics of Pref-CBR, which is essential for understanding our CBM strategy to be introduced in the next section. For further details of Pref-CBR, we refer to [10, 1, 2].

3.1 Basic Setting and Notation

Let a problem space $\mathbb{X}$ and a solution space $\mathbb{Y}$ be given. We assume $\mathbb{X}$ 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 Δ_Y.

In preference-based 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 over the solution space $\mathbb{Y}$. Here, we assume this class to consist of all linear order relations $\succ$ on $\mathbb{Y}$, and we denote the relation associated with a problem $\boldsymbol{x}$ by $\succ_{\boldsymbol{x}}$. More precisely, we assume that $\succ_{\boldsymbol{x}}$ has a specific form, which is defined by an "ideal" solution[3] $\boldsymbol{y}^* \in \mathbb{Y}$ 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, $\boldsymbol{y} \succ_{\boldsymbol{x}} \boldsymbol{z}$ iff $\Delta_Y(\boldsymbol{y}, \boldsymbol{y}^*) < \Delta_Y(\boldsymbol{z}, \boldsymbol{y}^*)$. 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.*

3.2 Case-based Inference

The key idea of preference-based 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 $\succ_{\boldsymbol{x}_0}$ or, equivalently, what is the most likely ideal solution $\boldsymbol{y}^*$ for $\boldsymbol{x}_0$?

We approach this problem from a statistical perspective, considering the true preference model $\succ_{\boldsymbol{x}_0}$ associated with the query $\boldsymbol{x}_0$ as a random variable 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

[3] The solution $\boldsymbol{y}^*$ could be a purely imaginary solution, which may not exist in practice.

$\mathcal{D}$ as a representative sample of $\mathbf{P}_\theta(\cdot)$ and, hence, estimating θ via maximum likelihood (ML) inference by

$$\theta^{ML} \;=\; \arg\max_{\theta\in\Theta} \mathbf{P}_\theta(\mathcal{D}) \;\; . \tag{1}$$

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. Our data generating process is based on the idea of a discrete choice model as used in choice and decision theory. More specifically, we assume the *logit* model of discrete choice:

$$\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^*) \;=\; \frac{1}{1+\exp\Big(-\beta\big(\Delta_Y(\boldsymbol{z},\boldsymbol{y}^*) - \Delta_Y(\boldsymbol{y},\boldsymbol{y}^*)\big)\Big)} \tag{2}$$

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 $\boldsymbol{y} \succ \boldsymbol{z}$. The coefficient β can be seen as a measure of precision of the preference feedback. For large β, $\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z})$ converges to 0 if $\Delta_Y(\boldsymbol{z},\boldsymbol{y}^*) < \Delta_Y(\boldsymbol{y},\boldsymbol{y}^*)$ and to 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 information source reporting preferences at random.

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 estimation principle. Assuming independence of the preferences, the likelihood of $\theta = (\boldsymbol{y}, \beta)$ is given by

$$\ell(\theta) = \ell(\theta \,|\, \mathcal{D}) = \prod_{i=1}^{N} \mathbf{P}\left(\boldsymbol{y}^{(i)} \succ \boldsymbol{z}^{(i)} \,|\, \theta\right) \;\; . \tag{3}$$

The ML estimation $\theta^{ML} = (\boldsymbol{y}^{ML}, \beta^{ML})$ of θ^* is given by the maximizer of (3):

$$\theta^{ML} = \left(\boldsymbol{y}^{ML}, \beta^{ML}\right) \;=\; \arg\max_{\boldsymbol{y}\in\mathbb{Y}_0,\, \beta\in\mathbb{R}_+} \ell(\boldsymbol{y},\beta) \tag{4}$$

The problem of finding this estimation in an efficient way is addressed in [10].

3.3 CBR as Preference-guided Search

Case-based inference as outlined above realizes a "one-shot prediction" of a promising solution for a query problem, given preferences in the context of similar problems encountered in the past. In a case-based problem solving process,

this prediction may thus serve as an initial solution, which is then adapted step by step. An adaptation process of that kind can be formalized as a search process, namely, a traversal of a suitable space of candidate solutions [3].

In the spirit of preference-based CBR, we implement case-based problem solving as a search process that is guided by preference information collected in previous problem solving episodes. To this end, 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 $\boldsymbol{y}$, e.g., by applying one of the available adaptation operators to $\boldsymbol{y}$.

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 $\boldsymbol{y} \succ_{\boldsymbol{x}_i} \boldsymbol{z}$, which 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. 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:

(i) Given a new query problem $\boldsymbol{x}_0$, the K nearest neighbors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K$ 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 $\mathcal{P}(\boldsymbol{x}_1), \ldots, \mathcal{P}(\boldsymbol{x}_K)$.

(ii) 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.

(iii) The search for a solution starts with an initial candidate $\boldsymbol{y}^\bullet \in \mathbb{Y}$, for example the "one-shot prediction" (4) based on $\mathcal{P}$, and iterates L times. Restricting the number of iterations by an upper bound L accounts for our assumption that an evaluation of a candidate solution is costly.

(iv) In each iteration, a new candidate $\boldsymbol{y}^{query}$ in the neighbourhood of $\boldsymbol{y}^\bullet$ is determined, based on (4) with $\mathbb{Y}_0 = \mathcal{N}(\boldsymbol{y}^\bullet)$, and given as a query to an (external) information source, which we refer to as the "oracle". Thus, the oracle is asked to compare $\boldsymbol{y}^{query}$ with the current best solution $\boldsymbol{y}^\bullet$. 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 $\boldsymbol{x}_0$, 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.

(v) When the search stops, the current best solution $\boldsymbol{y}^\bullet$ is returned (as an approximation of $\boldsymbol{y}^*$), 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 (iii) and (iv). Here, our case-based inference method is used to find the most promising candidate among the neighborhood of the current solution $\boldsymbol{y}^\bullet$, based on the preferences collected in the problem solving episode so far. 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.

4 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?

4.1 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 $\mathbb{Y}$:[4] It suggests moving toward those solutions $\boldsymbol{y}^*$ for which the probability $\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^*)$ in (2), 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 (3) 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.

[4] The notion of "direction" should not be taken literally. In fact, the mathematical structure of $\mathbb{Y}$ will normally not allow for defining a direction in a geometrical sense.

- **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 (2), 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.

4.2 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 (3) with $\mathbb{Y}_0 = \mathbb{Y}$. Such a comparison could easily be done using Δ_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.

Intra-case redundancy: Consider a case $(\boldsymbol{x}, \mathcal{P})$, and let $\boldsymbol{y}^\bullet$ denote the solution 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 (3) with and without the preference, i.e., the functions $\ell(\cdot \,|\, \mathcal{P})$ and $\ell(\cdot \,|\, \mathcal{P}')$ 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.

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 $\mathbf{P}(\boldsymbol{y} \succ \boldsymbol{z} \,|\, \boldsymbol{y}^\bullet) < 1/2$.

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 (3) of the preferences $\mathcal{P}_0$ under the associated (ML) parameters $(\boldsymbol{y}_0^\bullet, \beta)$ with the probability under $(\boldsymbol{y}_1^\bullet, \beta)$, i.e., when replacing $\boldsymbol{y}_1^\bullet$ by $\boldsymbol{y}_0^\bullet$. If

$$\frac{\ell(\boldsymbol{y}_1^\bullet, \beta \,|\, \mathcal{P}_0)}{\ell(\boldsymbol{y}_0^\bullet, \beta \,|\, \mathcal{P}_0)} \geq t \tag{5}$$

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

$$\Delta_Y(\boldsymbol{y}_0^\bullet, \boldsymbol{y}_1^\bullet) \leq v \tag{6}$$

for a second threshold $v \geq 0$. If both conditions are met, $(\boldsymbol{x}_0, \mathcal{P}_0)$ is considered redundant with respect to $(\boldsymbol{x}_1, \mathcal{P}_1)$.

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) is smaller than some (small) threshold or the distance in (6) 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.

4.3 Method

As a first step, we realized a "light" version of the above approach to maintenance in Pref-CBR by implementing the strategy for inter-case redundancy. More specifically, our strategy consists of the following steps:

- 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 [1] 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) and (6) are checked 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 at least one of the K cases, $(\boldsymbol{x}_0, \mathcal{P}_0)$ is considered redundant and not stored; otherwise, it is added to the case base **CB**.

Note that this strategy has three parameters, namely, the number of neighbours K and the thresholds t and v in (5–6).

5 Case Study

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 [12]; interestingly, the TSP problem has already been tackled by means of CBR by other authors [6, 7].
- The solution space $\mathbb{Y}$ is non-trivial but typically equipped with a natural structure, on which reasonable distance measures Δ_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.[5]

[5] 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

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.

5.1 Setting

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

- The problem space $\mathbb{X}$ is the set of all subsets $\boldsymbol{x} \subset \mathcal{X}$ of size $|\boldsymbol{x}| = 30$, where $\mathcal{X} \subset \mathbb{R}^2$ is a randomly created reference set of 75 points on the plane; 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 average squared distance between points in an optimal geometric superposition of the two point sets. This measure is computed using the "Procrustes Rotation" method, which is implemented in the "vegan" library of R,[6] subsequent to an optimal assignment of the points that is obtained by solving the linear assignment problem 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 $\{1, \ldots, 30\}$. This space is equipped with a local neighbourhood 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 following entries: path length, mean distance between each city to its nearest neighbors, standard deviation of distances of each city to its nearest neighbor. Then, the corresponding feature vectors are compared in term of their Euclidean distance.
- The parameters of Pref-CBR were set as follows: number of nearest neighbors $K = 17$, number of adaptation steps $L = 40$.

5.2 Experiment

In our experimental study, we compared Pref-CBR with and without case base maintenance. As additional baselines, we included a two random deletion policies, one that removes each newly observed case with a fixed probability (RCD) of 1/2, and one that removes individual preferences with the same fixed probability (RPD). We generated a sequence of 200 instances of the TSP problem

[6] http://cran.r-project.org/web/packages/vegan/index.html

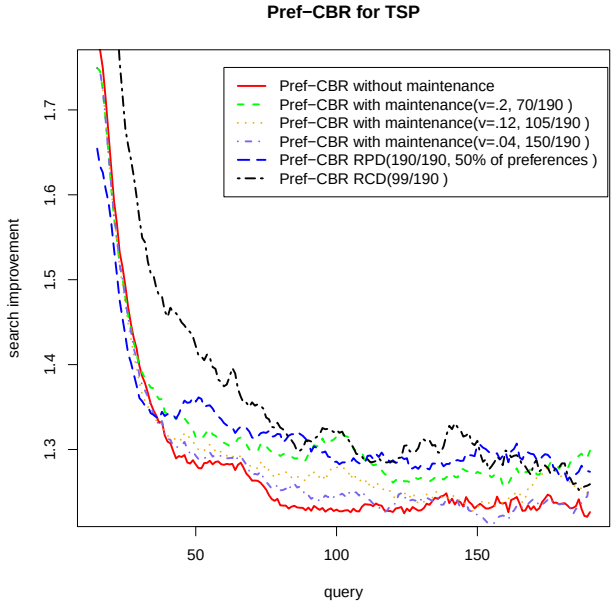

Fig. 1. Performance of Pref-CBR with and without maintenance on TSP data.

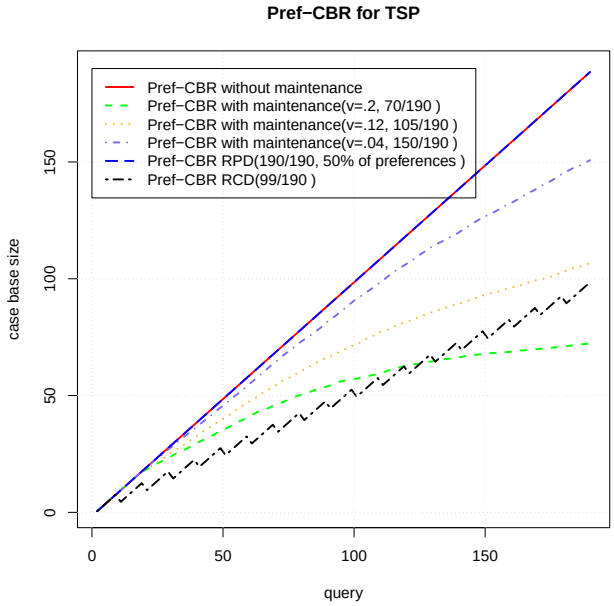

Fig. 2. Case base size for Pref-CBR search with and without maintenance of TSP data.

(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 1, both for Pref-CBR without maintenance and Pref-CBR with maintenance and different values of the parameter v (while the threshold t was fixed to 1). Moreover, Figure 2 shows the evolution of the size of the case base. As can be seen, the desired effect is indeed achieved: The size of the case base is significantly reduced while performance is maintained (in contrast to the random deletion policies). Moreover, the larger v, the stronger the tendency to delete cases. Thus, this parameter can be used to control the size of the case base.

6 Summary and Outlook

This paper extends our framework of preference-based CBR by a method for dynamic case base maintenance. The main goal of this method is to increase efficiency of case-based problem solving while maintaining performance. The effectiveness of our approach was illustrated in a case study with the traveling salesman problem.

So far, the implementation of our maintenance method includes only a part of the strategies discussed in Section 4, namely, a strategy for handling what we called inter-case redundancy. For future work, we therefore plan to extend this method by incorporating additional strategies for handling intra-case redundancy as well as intra- and inter-case noise.

References

1. A. Abdel-Aziz, W. Cheng, M. Strickert, and E. Hüllermeier. Preference-based CBR: A search-based problem solving framework. In S.J. Delany and S. Ontanon, editors, *Proceedings ICCBR–2013, 21th International Conference on Case-Based Reasoning*, pages 1–14, Saratoga Springs, NY, USA, 2013. Springer.
2. A. Abdel-Aziz, M. Strickert, and E. Hüllermeier. Learning solution similarity in preference-based CBR. In L. Lamontagne and E. Plaza, editors, *Proceedings ICCBR–2014, 22nd International Conference on Case-Based Reasoning*, pages 17–31, Cork, Ireland, 2014. Springer.
3. 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.
4. 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.

5. 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.
6. Padraig Cunningham, Barry Smyth, and Neil Hurley. On the use of CBR in optimisation problems such as the TSP. Technical Report TCD-CS-95-19, Trinity College Dublin, Department of Computer Science, 1995.
7. Hossein Erfani. Integrating case-based reasoning, knowledge-based approach and TSP algorithm for minimum tour finding. *Journal of Applied Mathematics Islamic Azad University of Lahijan*, 2006.
8. Geoffrey W. Gates. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, 18(3):431–433, 1972.
9. P. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(3):515–516, May 1968.
10. E. Hüllermeier and P. Schlegel. Preference-based CBR: First steps toward a methodological framework. In *Proceedings ICCBR–2011, 19th International Conference on Case-Based Reasoning*, pages 77–91. Springer-Verlag, 2011.
11. Vahid Jalali and David Leake. Adaptation-guided case base maintenance. In *Proc. AAAI, National Conference on Artificial Intelligence*, 2014.
12. 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.
13. 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.
14. Eduardo Lupiani, Jose M. Juarez, and Jose Palma. Evaluating case-base maintenance algorithms. *Knowledge-Based Systems*, 67:180 – 194, 2014.
15. 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*, volume 3201, pages 310–321. Springer-Verlag, 2004.
16. 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.
17. 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.
18. Abir Smiti and Zied Elouedi. Overview of maintenance for case based reasoning systems. *International Journal of Computer Applications*, 32(2):49–56, 2011.
19. B. Smyth and T. Keane. Remembering to forget. In C.S. Mellish, editor, *Proceedings International Joint Conference on Artificial Intelligence*, pages 377–382. Morgan Kaufmann, 1995.
20. Barry Smyth. Case-base maintenance. In Angel Pasqual del Pobil, Jos Mira, and Moonis Ali, editors, *Tasks and Methods in Applied Artificial Intelligence*, volume 1416 of *Lecture Notes in Computer Science*, pages 507–516. Springer Berlin Heidelberg, 1998.
21. 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.