Opportunistic Adaptation Knowledge Discovery

Fadi Badra[1], Amélie Cordier[2], and Jean Lieber[1]

[1] LORIA (CNRS, INRIA, Nancy Universities)
BP 239, 54506 Vandœuvre-lès-Nancy, France
{badra,lieber}@loria.fr
[2] LIRIS CNRS UMR 5202, Université Lyon 1, INSA Lyon, Université Lyon 2, ECL
43, bd du 11 novembre 1918, Villeurbanne, France
Amelie.Cordier@liris.cnrs.fr

Abstract. Adaptation has long been considered as the Achilles' heel of case-based reasoning since it requires some domain-specific knowledge that is difficult to acquire. In this paper, two strategies are combined in order to reduce the knowledge engineering cost induced by the adaptation knowledge (AK) acquisition task: AK is learned from the case base by the means of knowledge discovery techniques, and the AK acquisition sessions are opportunistically triggered, i.e., at problem-solving time.

1 Introduction

Case-based reasoning (CBR [6]) is a reasoning paradigm based on the reuse of previous problem-solving experiences, called cases. A CBR system often has profit of a retrieval procedure, selecting in a case base a source case similar to the target problem, and an adaptation procedure, that adapts the retrieved source case to the specificity of the target problem. The adaptation procedure depends on domain-dependent adaptation knowledge (AK, in the following). Acquiring AK can be done from experts or by using machine learning techniques. An intermediate approach is knowledge discovery (KD) that combines efficient learning algorithms with human-machine interaction.

Most of previous AK acquisition strategies are off-line: they are disconnected from the use of the CBR system. By contrast, recent work aims at integrating AK acquisition from experts to specific reasoning sessions: this *opportunistic* AK acquisition takes advantage of the problem-solving context. This paper presents an approach to AK discovery that is opportunistic: the KD is triggered at problem-solving time.

The paper is organized as follows. Section 2 introduces some basic notions and notations about CBR. Section 3 presents the CBR system TAAABLE, which constitutes the application context of the study, and motivates the need for adaptation knowledge acquisition in this application context. Section 4 presents the proposed opportunistic and interactive AK discovery method. In Sect. 5, this method is applied to acquire adaptation knowledge in the context of the TAAABLE system. Section 6 discusses this approach and situates it among related work. Section 7 concludes and presents some future work.

2 Basic Notions About CBR

In the following, problems are assumed to be represented in a language $\mathcal{L}_{\mathtt{pb}}$ and solutions in a language $\mathcal{L}_{\mathtt{sol}}$. A *source case* represents a problem-solving episode by a pair $(\mathtt{srce}, \mathtt{Sol}(\mathtt{srce}))$, in which $\mathtt{srce} \in \mathcal{L}_{\mathtt{pb}}$ is the representation of a problem statement and $\mathtt{Sol}(\mathtt{srce}) \in \mathcal{L}_{\mathtt{sol}}$ is the representation of its associated solution. CBR aims at solving a *target problem* $\mathtt{tgt}$ using a set of source cases $\mathtt{CB}$ called the *case base*. The CBR process is usually decomposed in two main steps: retrieval and adaptation. *Retrieval* selects a source case $(\mathtt{srce}, \mathtt{Sol}(\mathtt{srce}))$ from the case base such that $\mathtt{srce}$ is judged to be similar to $\mathtt{tgt}$ according to a given similarity criterion. *Adaptation* consists in modifying $\mathtt{Sol}(\mathtt{srce})$ in order to propose a candidate solution $\widetilde{\mathtt{Sol}(\mathtt{tgt})}$ for $\mathtt{tgt}$ to the user. If the user validates the candidate solution $\widetilde{\mathtt{Sol}(\mathtt{tgt})}$, then $\widetilde{\mathtt{Sol}(\mathtt{tgt})}$ is considered to be a solution $\mathtt{Sol}(\mathtt{tgt})$ for $\mathtt{tgt}$.

3 Application Context: The TAAABLE System

The TAAABLE system [3] is a cooking CBR system. In the cooking domain, CBR aims at answering a query using a set of recipes. In order to answer a query, the system retrieves a recipe in the recipe set and adapts it to produce a recipe satisfying the query. The TAAABLE system was proposed to participate to the *Computer Cooking Contest* (CCC) challenge in 2008 [4]. In the CCC challenge, queries are given in natural language and express a set of constraints that the desired recipe should satisfy. These constraints concern the ingredients to be included or avoided, the type of ingredients (e.g., meat or fruit), the dietary practice (e.g., nut-free diet), the type of meal (e.g., soup) or the type of cuisine (e.g., chinese cuisine). An example of query is: "Cook a chinese soup with leek but no peanut oil." Recipes are given in textual form, with a shallow XML structure, and include a set of ingredients together with a textual part describing the recipe preparation. The TAAABLE system is accessible online (`http://taaable.fr`).

3.1 Representation Issues

A Cooking Ontology. The system makes use of a cooking ontology $\mathcal{O}$ represented in propositional logic. Each concept of $\mathcal{O}$ corresponds to a propositional variable taken from a finite set $\mathcal{V}$ of propositional variables. $\mathcal{O}$ is mainly composed of a set of concepts organized in a hierarchy, which corresponds, in propositional logic, to a set of logical implications $\mathtt{a} \Rightarrow \mathtt{b}$. For example, the axiom $\mathtt{leek} \Rightarrow \mathtt{onions}$ of $\mathcal{O}$ states that leeks are onions.

Problem and Solution Representation. In TAAABLE, a problem $\mathtt{pb} \in \mathcal{L}_{\mathtt{pb}}$ represents a query and a solution $\mathtt{Sol}(\mathtt{pb})$ of $\mathtt{pb}$ represents a recipe that matches this query. $\mathcal{L}_{\mathtt{pb}}$ and $\mathcal{L}_{\mathtt{sol}}$ are chosen fragments of propositional logic defined using the vocabulary $\mathcal{V}$ introduced in the cooking ontology $\mathcal{O}$. One propositional variable is defined in $\mathcal{L}_{\mathtt{pb}}$ and $\mathcal{L}_{\mathtt{sol}}$ for each concept name of $\mathcal{O}$ and the only logical connective used in $\mathcal{L}_{\mathtt{pb}}$ and

$\mathcal{L}_{\texttt{sol}}$ is the conjunction $\wedge$. For example, the representation $\texttt{tgt} \in \mathcal{L}_{\texttt{pb}}$ of the query mentioned above is:

$$\texttt{tgt} = \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{leek} \wedge \neg\,\texttt{peanut\_oil}$$

The case base $\texttt{CB}$ contains a set of recipes. Each recipe is indexed in the case base by a propositional formula $\texttt{R} \in \mathcal{L}_{\texttt{sol}}$. For example, the index $\texttt{R}$ of the recipe *Wonton Soup* is:

$$\texttt{R} = \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{green\_onion} \wedge \ldots \wedge \texttt{peanut\_oil} \wedge \textit{Nothing else}$$

Nothing else denotes a conjunction of negative literals $\neg\,\texttt{a}$ for all $\texttt{a} \in \mathcal{V}$ such that $\texttt{chinese} \wedge \texttt{soup} \wedge \texttt{green\_onion} \wedge \ldots \wedge \texttt{peanut\_oil} \nvDash_{\mathcal{O}} \texttt{a}$. This kind of "closed world assumption" states explicitly that for all propositional variable $\texttt{a} \in \mathcal{V}$, either $\texttt{R} \vDash_{\mathcal{O}} \texttt{a}$ (the recipe contains the ingredient represented by $\texttt{a}$) or $\texttt{R} \vDash_{\mathcal{O}} \neg\,\texttt{a}$ (the recipe does not contain the ingredient represented by $\texttt{a}$).

Each recipe index $\texttt{R}$ represents a set of source cases: $\texttt{R}$ represents the set of source cases $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ such that $\texttt{Sol}(\texttt{srce}) = \texttt{R}$ and $\texttt{srce}$ is solved by $\texttt{R}$, i.e., $\texttt{srce}$ is such that $\texttt{R} \vDash_{\mathcal{O}} \texttt{srce}$.

Adaptation Knowledge. In TAAABLE, adaptation knowledge is given by a set of reformulations $(\texttt{r}, \mathcal{A}_{\texttt{r}})$ in which $\texttt{r}$ is a binary relation between problems and $\mathcal{A}_{\texttt{r}}$ is an adaptation function associated with $\texttt{r}$ [13]. A reformulation has the following semantics: if two problems $\texttt{pb}_1$ and $\texttt{pb}_2$ are related by $\texttt{r}$ —denoted by $\texttt{pb}_1\ \texttt{r}\ \texttt{pb}_2$— then for every recipe $\texttt{Sol}(\texttt{pb}_1)$ matching the query $\texttt{pb}_1$, $\mathcal{A}_{\texttt{r}}(\texttt{pb}_1, \texttt{Sol}(\texttt{pb}_1), \texttt{pb}_2) = \widetilde{\texttt{Sol}}(\texttt{pb}_2)$ matches the query $\texttt{pb}_2$.

In this paper, binary relations $\texttt{r}$ are given by substitutions of the form $\sigma = \alpha \rightsquigarrow \beta$, where α and β are literals (either positive or negative). For example, the substitution $\sigma = \texttt{leek} \rightsquigarrow \texttt{onions}$ generalizes leek into onions.

Adaptation functions $\mathcal{A}_{\texttt{r}}$ are given by substitutions of the form $\Sigma = \texttt{A} \rightsquigarrow \texttt{B}$ in which $\texttt{A}$ and $\texttt{B}$ are conjunctions of literals. For example, the substitution $\Sigma = \texttt{soup} \wedge \texttt{pepper} \rightsquigarrow \texttt{soup} \wedge \texttt{ginger}$ states that pepper can be replaced by ginger in soup recipes. A substitution Σ can be automatically generated from a substitution σ: $\Sigma = \texttt{b} \rightsquigarrow \texttt{a}$ if σ is of the form $\texttt{a} \rightsquigarrow \texttt{b}$ and $\Sigma = \emptyset \rightsquigarrow \neg\,\texttt{a}$ if σ is of the form $\neg\,\texttt{a} \rightsquigarrow \emptyset$.

The main source of adaptation knowledge is the ontology $\mathcal{O}$. A substitution $\sigma = \texttt{a} \rightsquigarrow \texttt{b}$ is automatically generated from each axiom $\texttt{a} \Rightarrow \texttt{b}$ of $\mathcal{O}$ and correspond to a *substitution by generalization.* A substitution $\sigma = \texttt{a} \rightsquigarrow \texttt{b}$ can be applied to a query $\texttt{pb}$ if $\texttt{pb} \vDash_{\mathcal{O}} \texttt{a}$. σ generates a new query $\sigma(\texttt{pb})$ in which the propositional variable $\texttt{a}$ has been substituted by the propositional variable $\texttt{b}$. For example, the substitution $\sigma = \texttt{leek} \rightsquigarrow \texttt{onions}$ is generated automatically from the axiom $\texttt{leek} \Rightarrow \texttt{onions}$ of $\mathcal{O}$. σ can be applied to the query $\texttt{tgt}$ to produce the query $\sigma(\texttt{tgt}) = \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{onions} \wedge \neg\,\texttt{peanut\_oil}$, in which $\texttt{leek}$ has been substituted by $\texttt{onions}$. For each propositional variable $\texttt{a}$ of $\mathcal{V}$, an additional substitution of the form $\sigma = \neg\,\texttt{a} \rightsquigarrow \emptyset$ is generated. Such a substitution can be applied to a problem $\texttt{pb}$ if $\texttt{pb} \vDash_{\mathcal{O}} \neg\,\texttt{a}$ and generates a new problem $\sigma(\texttt{pb})$ in which the negative literal $\neg\,\texttt{a}$ is removed. This has the effect to loosen the constraints imposed on a query e.g., by omitting in the query an unwanted ingredient. For example, the substitution $\neg\,\texttt{peanut\_oil} \rightsquigarrow \emptyset$ applied to tgt generates the query $\sigma(\texttt{tgt}) = \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{leek}$, in which the condition on the ingredient peanut_oil is omitted.

However, when $\mathcal{O}$ is the only source of adaptation knowledge, the system is only able to perform simple adaptations, in which the modifications made to Sol(srce) correspond to a sequence of substitutions that can be used to transform srce into tgt. Therefore, an additional adaptation knowledge base AKB is introduced. AKB contains a set of reformulations (σ, Σ) that capture more complex adaptation strategies.

3.2 The CBR Process in Taaable

Retrieval. The retrieval algorithm is based on a *smooth classification* algorithm on an index hierarchy. Such an algorithm aims at determining a set of modifications to apply to tgt in order to obtain a modified query srce that matches at least one recipe Sol(srce) of the case base. The algorithm computes a *similarity path*, which is a composition of substitutions $\texttt{SP} = \sigma_q \circ \sigma_{q-1} \circ \cdots \circ \sigma_1$ such that there exists at least one recipe Sol(srce) matching the modified query $\texttt{srce} = \sigma_q(\sigma_{q-1}(\ldots\sigma_1(\texttt{tgt})\ldots))$, i.e., such that $\texttt{Sol(srce)} \vDash_{\mathcal{O}} \texttt{srce}$ holds. Thus, a similarity path SP can be written:

$$\texttt{Sol(srce)} \vDash_{\mathcal{O}} \texttt{srce} \xleftarrow{\sigma_q} \xleftarrow{\sigma_{q-1}} \cdots \xleftarrow{\sigma_1} \texttt{tgt}$$

For example, to solve the above query tgt, the system generates a similarity path $\texttt{SP} = \sigma_2 \circ \sigma_1$, with:

$$\begin{aligned}
\texttt{tgt} &= \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{leek} \wedge \neg\texttt{peanut\_oil}\\
\sigma_1 &= \neg\texttt{peanut\_oil} \rightsquigarrow \emptyset, \quad \sigma_2 = \texttt{leek} \rightsquigarrow \texttt{onions}\\
\texttt{srce} &= \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{onions}\\
\texttt{Sol(srce)} &= \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{green\_onion} \wedge \ldots \wedge \texttt{peanut\_oil} \wedge \textit{Nothing else}
\end{aligned}$$

In this similarity path, Sol(srce) is the propositional representation of the recipe *Wonton Soup*. Since the ontology $\mathcal{O}$ contains the axiom $\texttt{green\_onion} \Rightarrow \texttt{onions}$, the modified query $\texttt{srce} = \sigma_2 \circ \sigma_1\,(\texttt{tgt})$ verifies $\texttt{Sol(srce)} \vDash_{\mathcal{O}} \texttt{srce}$.

Adaptation. To a similarity path is associated an *adaptation path* AP, which is a composition of substitutions $\texttt{AP} = \Sigma_1 \circ \Sigma_2 \circ \cdots \circ \Sigma_q$ such that the modified recipe $\widetilde{\texttt{Sol}}\texttt{(tgt)} = \Sigma_1(\Sigma_2(\ldots\Sigma_q(\texttt{Sol(srce)})\ldots))$ solves the initial query tgt, i.e., verifies $\widetilde{\texttt{Sol}}\texttt{(tgt)} \vDash_{\mathcal{O}} \texttt{tgt}$. Thus, an adaptation path AP can be written

$$\texttt{Sol(srce)} \xrightarrow{\Sigma_q} \xrightarrow{\Sigma_{q-1}} \cdots \xrightarrow{\Sigma_1} \widetilde{\texttt{Sol}}\texttt{(tgt)} \vDash_{\mathcal{O}} \texttt{tgt}$$

The adaptation path AP is constructed from the similarity path SP by associating a substitution Σ_i to each substitution σ_i. To determine which substitution Σ_i to associate to a given substitution σ_i, the external adaptation knowledge base AKB is searched first. For a substitution $\sigma_i = \alpha \rightsquigarrow \beta$, the system looks for a substitution $\Sigma = \texttt{A} \rightsquigarrow \texttt{B}$ such that $\texttt{A} \vDash_{\mathcal{O}} \beta$ and $\texttt{B} \vDash_{\mathcal{O}} \alpha$. For example, if $\sigma_2 = \texttt{leek} \rightsquigarrow \texttt{onions}$ is used in SP and AKB contains the reformulation (σ, Σ) with $\sigma = \sigma_2$ and $\Sigma = \texttt{green\_onion} \rightsquigarrow \texttt{leek} \wedge \texttt{ginger}$, Σ will be selected to constitute the substitution Σ_2 in AP since

$$\begin{tikzcd}
\mathtt{srce} \arrow[r, dash, "\sigma_2"] \arrow[d] & \mathtt{pb} \arrow[r, dash, "\sigma_1"] \arrow[d] & \mathtt{tgt} \arrow[d] \\
\mathtt{Sol(srce)} \arrow[r, dash, "\Sigma_2"'] & \widetilde{\mathtt{Sol}}\mathtt{(pb)} \arrow[r, dash, "\Sigma_1"'] & \widetilde{\mathtt{Sol}}\mathtt{(tgt)}
\end{tikzcd}$$

Fig. 1. A similarity path and the associated adaptation path

$\mathtt{green\_onion} \vDash_{\mathcal{O}} \mathtt{onions}$ and $\mathtt{leek} \wedge \mathtt{ginger} \vDash_{\mathcal{O}} \mathtt{leek}$. If no substitution Σ is found in AKB for a given substitution σ_i then Σ_i is generated automatically from σ_i.

In the previous example, AKB is considered to be empty so Σ_1 and Σ_2 are generated automatically from the substitutions σ_1 and σ_2: $\Sigma_1 = \emptyset \rightsquigarrow \neg\mathtt{peanut\_oil}$ since $\sigma_1 = \neg\mathtt{peanut\_oil} \rightsquigarrow \emptyset$ and $\Sigma_2 = \mathtt{onions} \rightsquigarrow \mathtt{leek}$ since $\sigma_2 = \mathtt{leek} \rightsquigarrow \mathtt{onions}$. According to the axiom $\mathtt{green\_onion} \Rightarrow \mathtt{onions}$ of $\mathcal{O}$, the system further specializes the substitution Σ_2 into the substitution $\mathtt{green\_onion} \rightsquigarrow \mathtt{leek}$ and the user is proposed to replace green onions by leek in the recipe *Wonton Soup* and to suppress peanut oil. The generated adaptation path is $\mathtt{AP} = \Sigma_1 \circ \Sigma_2$ (Fig. 1), with:

$$\begin{aligned}
\mathtt{Sol(srce)} &= \mathtt{chinese} \wedge \mathtt{soup} \wedge \mathtt{green\_onion} \wedge \ldots \wedge \mathtt{peanut\_oil} \wedge \textit{Nothing else} \\
\Sigma_2 &= \mathtt{green\_onion} \rightsquigarrow \mathtt{leek}, \quad \Sigma_1 = \emptyset \rightsquigarrow \neg\mathtt{peanut\_oil} \\
\widetilde{\mathtt{Sol}}\mathtt{(tgt)} &= \mathtt{chinese} \wedge \mathtt{soup} \wedge \mathtt{leek} \wedge \ldots \wedge \neg\mathtt{peanut\_oil} \wedge \textit{Nothing else} \\
\mathtt{tgt} &= \mathtt{chinese} \wedge \mathtt{soup} \wedge \mathtt{leek} \wedge \neg\mathtt{peanut\_oil}
\end{aligned}$$

The inferred solution $\widetilde{\mathtt{Sol}}\mathtt{(tgt)}$ solves the initial query $\mathtt{tgt}$: $\widetilde{\mathtt{Sol}}\mathtt{(tgt)} \vDash_{\mathcal{O}} \mathtt{tgt}$.

3.3 Why Learning Adaptation Knowledge in TAAABLE?

In the version of the TAAABLE system that was proposed to participate in the CCC challenge, $\mathtt{AKB} = \emptyset$ so adaptation knowledge is inferred from the ontology $\mathcal{O}$. The main advantage of this approach lies in its simplicity: no external source of adaptation knowledge is needed and the system is able to propose a solution to any target problem. However, the system's adaptation capabilities (simple substitutions) appear to be very limited and the user has no means to give some feedback on the quality of the proposed adaptation.

For example, the substitution $\Sigma_1 = \emptyset \rightsquigarrow \neg\mathtt{peanut\_oil}$ suggests to remove the ingredient peanut oil in the retrieved recipe, but as the oil is used in this recipe to saute the bok choy, the adapted recipe turns out to be practically unfeasible. A better adaptation would suggest to replace peanut oil by e.g., sesame oil, which can be modeled by the substitution $\Sigma_1 = \mathtt{peanut\_oil} \rightsquigarrow \mathtt{sesame\_oil}$. To generate this substitution automatically, the system could for example exploit the fact that the concepts $\mathtt{peanut\_oil}$ and $\mathtt{sesame\_oil}$ are both sub-concepts of the concept $\mathtt{oil}$ in $\mathcal{O}$. But still, some additional knowledge would be needed to express the fact that peanut oil should be replaced by sesame oil, and not by olive oil or hot chili oil, as $\mathtt{olive\_oil}$ and $\mathtt{hot\_chili\_oil}$ are also sub-concepts of $\mathtt{oil}$ in $\mathcal{O}$. Besides, the system should be aware that this substitution is recommended *only in Asian cuisine*, which can be modeled by the more precise substitution $\Sigma_1 = \texttt{asian} \wedge \texttt{peanut\_oil} \rightsquigarrow \texttt{asian} \wedge \texttt{sesame\_oil}$.

Furthermore, the second substitution $\Sigma_2 = \texttt{green\_onions} \rightsquigarrow \texttt{leek}$ suggests to solely replace sliced green onions by uncooked leek. But the green onion was used in the original *Wonton Soup* for garniture, so the user might consider that raw leek added as garniture alters too much the taste of a soup. A better adaptation would consist in frying leek with e.g., tempeh and red bell pepper to prepare the garniture. Such an adaptation can be modeled by the substitution $\Sigma_2 = \texttt{green\_onions} \rightsquigarrow \texttt{leek} \wedge \texttt{tempeh} \wedge \texttt{red\_bell\_pepper}$. This substitution, which reflects a cooking know-how, can hardly be generated automatically from the ontology.

These examples show that in order to improve its adaptation capabilities, the system would greatly benefit from the availability of a set of adaptation rules that would capture more complex adaptation strategies. These adaptation rules cannot be generated automatically from the ontology and need to be acquired from other knowledge sources. These examples also show that the human expert plays a major role in adaptation knowledge acquisition and that in the cooking domain, adaptation rules are often highly contextual.

4 Opportunistic Adaptation Knowledge Discovery

The presented AK acquisition method combines two previous approaches of AK acquisition. The first one was implemented in the CABAMAKA system [5] and learns AK from differences between cases by the means of knowledge discovery techniques (section 4.1). The second one was implemented in the IAKA system [8] and acquires adaptation knowledge at problem-solving time through interactions with the user (section 4.2).

4.1 Adaptation Knowledge Discovery from the Case Base

Machine learning algorithms aim at extracting some regularities from a set of observations. Knowledge discovery techniques combine efficient machine learning algorithms with human-machine interaction. In [5], AK is learned from differences between cases by the means of knowledge discovery techniques. A set of pairs of sources cases is taken as input of a frequent itemset extraction algorithm, which outputs a set of itemsets. Each of these itemsets can be interpreted as an adaptation rule. This approach of AK learning was motivated by the original idea proposed by Kathleen Hanney and Mark T. Keane in [11], in which the authors suggest that AK may be learned from differences between cases. The main assumption is that the differences that occur between cases in the case base are often representative of differences that will occur between future problems and the case base.

To learn adaptation rules from differences between cases, representing variations between cases is essential. In [2], expressive representation formalisms are proposed and it is shown that defining a partial order on the variation language can help organizing the learned rules by generality.

4.2 Opportunistic and Interactive Knowledge Acquisition

Experiential knowledge, or know-how, can often be acquired on-line, when users are using CBR tools. It is the aim of interactive and opportunistic knowledge acquisition strategies to support such an acquisition. In these strategies, the system exploits its interactions with its user to build new pieces of knowledge, to test them and, in case of success, to retain them. Moreover, the knowledge acquisition process is often opportunistic, i.e, triggered by a previous reasoning failure: reasoning failures highlight missing knowledge and thus constitute a guidance for the acquisition process. A major advantage of interactive knowledge acquisition strategies is that they ensure that the user is in a favorable context when he participates to the acquisition process. In [7], a review of interactive and opportunistic knowledge acquisition approaches is proposed, and two strategies are developed. This work illustrates the efficiency of interactive and opportunistic knowledge acquisition approaches to acquire specific knowledge. On the other hand, it shows that such approaches only allow the systems to acquire small pieces of knowledge at a time.

4.3 Combining the Two Approaches

When properly used, knowledge discovery techniques may have the strong advantage of automating a part of the knowledge acquisition process. In these approaches, dedicated human-machine interfaces allow the expert, through predefined interactions, to provide feedback on a set of suggestions generated automatically by the system. The role of the expert is thus reduced to the validation of a pre-selected set of knowledge pieces. The acquired knowledge is directly usable by the system, without the need for an additional formalization step. Automatic approaches also benefit from efficient machine learning algorithms that can be applied, as in [2], to learn adaptation rules at different levels of generality. However, these approaches still produce a large number of candidate knowledge units that have to be validated by a domain expert out of any context, which constitutes an important drawback.

Acquiring adaptation knowledge *offline*, i.e., independently of a particular problem-solving session, appears to be problematic. Offline AK acquisition forces the system's designer to anticipate the need for adaptation knowledge in problem-solving and to acquire it in advance, which can be very tedious, if not impossible. Offline acquisition of adaptation knowledge also makes difficult to come up with fine-grained adaptation rules, since adaptation knowledge is often highly contextual. For example, in the cooking domain, an egg can sometimes be substituted by 100 grams of tofu, but this adaptation rule may be applied only to certain types of dishes, like cakes or mayonnaise, and has proved to be irrelevant in order to adapt a mousse recipe or an omelet recipe. Acquiring such a rule would require to circumscribe its domain of validity in order to avoid over-generalization.

Moreover, initial acquisition of adaptation knowledge prevents the system from learning from experience. A CBR system with fixed adaptation knowledge has no way to improve its problem-solving capabilities, except by retaining in the case base a new experience each time a problem has been solved, as it is usually done in traditional CBR systems [6].

On the other hand, interactive and opportunistic knowledge acquisition approaches heavily rely on the human expert but ensure that the expert is "in context" when validating knowledge units that are to be acquired. Combining knowledge discovery techniques and interactive approaches, as it is proposed here, could overcome one of the limitations of KD by dramatically reducing the number of candidate adaptation rules presented to the expert. By triggering the process in an opportunistic manner, the expert is able to parametrize the KD in order to focus on specific knowledge to acquire in context. The resulting AK discovery process:

- is performed *on-line*, i.e., in the context of a problem-solving session,
- is *interactive* as adaptation knowledge is learned by the system through interactions with its user who acts as an expert,
- is *opportunistic* as it is triggered by reasoning failure, and, consequently, often helps repairing a failed adaptation,
- makes use of knowledge discovery techniques to provide *assistance* to the user in the formulation of new knowledge: the user is presented with a set of suggestions that are generated automatically from the case base.

5 Applying Opportunistic AK Discovery to TAABLE

In this section, an opportunistic AK discovery process is applied to the context of the TAABLE system.

5.1 AK Discovery

In TAABLE, the AK discovery process consists in learning a set of substitutions from the case base by comparing two sets of recipes.

The Training Set. The training set $\mathtt{TS}$ is formed by selecting from the case base a set of pairs of recipes $(\mathtt{R}_k, \mathtt{R}_\ell) \in \mathtt{CB} \times \mathtt{CB}$ and by representing for each selected pair of recipes $(\mathtt{R}_k, \mathtt{R}_\ell)$ the variation $\Delta_{k\ell}$ from $\mathtt{R}_k$ to $\mathtt{R}_\ell$. The choice of the training set $\mathtt{TS}$ results from a set of interactions with the user during which he/she is asked to formulate the cause of the adaptation failure and to pick up a repair strategy.

Representing Variations. The variation $\Delta_{k\ell}$ from a recipe $\mathtt{R}_k$ to a recipe $\mathtt{R}_\ell$ is represented in a language $\mathcal{L}_\Delta$ by a set of properties. Three properties $\mathtt{a}^-$, $\mathtt{a}^+$ and $\mathtt{a}^=$ are defined in $\mathcal{L}_\Delta$ for each propositional variable $\mathtt{a}$ of $\mathcal{V}$, and $\Delta_{k\ell} \in \mathcal{L}_\Delta$ contains:

- the property $\mathtt{a}^-$ if $\mathtt{R}_k \vDash_{\mathcal{O}} \mathtt{a}$ and $\mathtt{R}_\ell \nvDash_{\mathcal{O}} \mathtt{a}$,
- the property $\mathtt{a}^+$ if $\mathtt{R}_k \nvDash_{\mathcal{O}} \mathtt{a}$ and $\mathtt{R}_\ell \vDash_{\mathcal{O}} \mathtt{a}$,
- the property $\mathtt{a}^=$ if $\mathtt{R}_k \vDash_{\mathcal{O}} \mathtt{a}$ and $\mathtt{R}_\ell \vDash_{\mathcal{O}} \mathtt{a}$.

For example, if:

$$\mathtt{R}_k = \mathtt{chinese} \wedge \mathtt{soup} \wedge \ldots \wedge \mathtt{peanut\_oil} \wedge \textit{Nothing else}$$
$$\mathtt{R}_\ell = \mathtt{chinese} \wedge \mathtt{soup} \wedge \ldots \wedge \mathtt{olive\_oil} \wedge \textit{Nothing else}$$

then $\Delta_{k\ell} = \{\texttt{chinese}^=, \texttt{soup}^=, \texttt{oil}^=, \texttt{peanut\_oil}^-, \texttt{olive\_oil}^+, \ldots\}$, provided that $\texttt{peanut\_oil} \models_O \texttt{oil}$, $\texttt{olive\_oil} \models_O \texttt{oil}$, $\texttt{R}_\ell \not\models_O \texttt{peanut\_oil}$ and $\texttt{R}_k \not\models_O \texttt{olive\_oil}$.

The inclusion relation $\subseteq$ constitutes a partial order on $\mathcal{L}_\Delta$ that can be used to organize variations by generality: a variation Δ is more general than a variation Δ' if $\Delta \subseteq \Delta'$.

Mining. The learning process consists in highlighting some variations $\Delta \in \mathcal{L}_\Delta$ that are more general than a "large" number of elements $\Delta_{k\ell}$ of TS. More formally, let

$$\texttt{support}(\Delta) = \frac{\texttt{card}\ \{\Delta_{k\ell} \in \texttt{TS} \mid \Delta \subseteq \Delta_{k\ell}\}}{\texttt{card TS}}$$

Learning adaptation rules aims at finding the $\Delta \in \mathcal{L}_\Delta$ such that $\texttt{support}(\Delta) \geq \sigma_s$, where $\sigma_s \in [0;1]$ is a learning parameter called the support threshold. It can be noticed that if $\Delta_1 \subseteq \Delta_2$ then $\texttt{support}(\Delta_1) \geq \texttt{support}(\Delta_2)$. The support threshold also has an influence on the number of generated variations. The number of generated variations increases when σ_s decreases. Thus, specifying a high threshold restricts the generation of variations to the most general ones, which can limit the number of generated variations and save computation time but has the effect to discard the most specific ones from the result set.

Each learned variation $\Delta = \{\texttt{p}_1, \texttt{p}_2, \ldots, \texttt{p}_n\} \in \mathcal{L}_\Delta$ is interpreted as a substitution of the form $\texttt{A} \rightsquigarrow \texttt{B}$ such that:

- $\texttt{A} \models_O \texttt{a}$ and $\texttt{B} \not\models_O \texttt{a}$ if $\texttt{a}^- \in \Delta$,
- $\texttt{A} \not\models_O \texttt{a}$ and $\texttt{B} \models_O \texttt{a}$ if $\texttt{a}^+ \in \Delta$,
- $\texttt{A} \models_O \texttt{a}$ and $\texttt{B} \models_O \texttt{a}$ if $\texttt{a}^= \in \Delta$.

For example, the variation $\Delta = \{\texttt{oil}^=, \texttt{peanut\_oil}^-, \texttt{olive\_oil}^+\}$ is interpreted as the substitution $\Sigma = \texttt{peanut\_oil} \rightsquigarrow \texttt{olive\_oil}$. The conjunct oil is not present neither in A nor in B since it is useless: $\texttt{peanut\_oil} \models_O \texttt{oil}$ and $\texttt{olive\_oil} \models_O \texttt{oil}$.

Filtering. For a retrieved recipe Sol(srce), the result set can be filtered in order to retain only the substitutions $\Sigma = \texttt{A} \rightsquigarrow \texttt{B}$ that can be applied to modify Sol(srce), i.e., such that $\texttt{Sol(srce)} \models_O \texttt{A}$.

Validation. Knowledge discovery aims at building a model of reality from a set of observations. But as a model of a part of reality is only valid with respect to a particular observer, any learned substitution has to be validated by a human expert in order to acquire the status of piece of knowledge.

5.2 Opportunistic Adaptation Knowledge Discovery

The AK discovery process turns the case base into an additional source of adaptation knowledge. This new source of knowledge is used during a problem-solving session to provide the CBR system with adaptation knowledge "on demand". A set of variations Δ is learned from the case base by comparing two sets of recipes and each learned variation Δ is interpreted as a substitution Σ that can be used to repair the adaptation path $\mathtt{AP}$. Each learned substitution Σ is presented to the user for validation together with the corrected solution $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$ resulting from its application. When the user validates the corrected solution, a new reformulation (σ, Σ) is added to the adaptation knowledge base $\mathtt{AKB}$ so that the learned substitution Σ can be later reused to adapt new recipes. The AK discovery process is triggered either during the adaptation phase, to come up with suggestions of gradual solution refinements (see section 5.4 for an example), or during the solution test phase to repair a failed adaptation in response to the user's feedback (see section 5.5 for an example).

5.3 Implementation

To test the proposed adaptation knowledge acquisition method, a prototype was implemented that integrates the TAAABLE system [3] and the CABAMAKA system [5]. The case base contains 862 recipes taken from the CCC 2008 recipe set. The TAAABLE system is used to perform retrieval and adaptation. The CABAMAKA system is used to learn a set of substitutions Σ from the case base from the comparison of two sets of recipes. As in [5], the mining step is performed thanks to a frequent closed itemset extraction algorithm.

5.4 A First Example: Cooking a Chocolate Cake

An example is presented to illustrate how the case base is used as an additional source of adaptation knowledge. The AK discovery process is parametrized automatically and is used to provide assistance to the user by suggesting some gradual refinements for the proposed solution.

1. *Representing the Target Problem.* In this example, the user wants to cook a chocolate cake with baking chocolate and oranges. The target problem is:

$$\mathtt{tgt} = \mathtt{cake} \wedge \mathtt{baking\_chocolate} \wedge \mathtt{orange}$$

In the TAAABLE interface, the field "Ingredients I Want" is filled in with the tokens `baking_chocolate` and `orange` and the field "Types I Want" is filled in with the token `cake`.

2. *Retrieval.* The retrieval procedure generates the similarity path $\mathtt{SP} = \sigma_1$ in which the substitution $\sigma_1 = \mathtt{baking\_chocolate} \rightsquigarrow \mathtt{chocolate}$ is generated automatically from the ontology $\mathcal{O}$ from the axiom $\mathtt{baking\_chocolate} \Rightarrow \mathtt{chocolate}$. $\mathtt{SP}$ is applied to $\mathtt{tgt}$ in order to produce the modified query $\mathtt{srce} = \mathtt{cake} \wedge \mathtt{chocolate} \wedge \mathtt{orange}$. The system retrieves the recipe *Ultralight Chocolate Cake*, whose representation $\mathtt{Sol}(\mathtt{srce})$ is:

$$\mathtt{Sol}(\mathtt{srce}) = \mathtt{cake} \wedge \mathtt{cocoa} \wedge \mathtt{orange} \wedge \ldots \wedge \textit{Nothing else}$$

Since the ontology $\mathcal{O}$ contains the axiom $\mathtt{cocoa} \Rightarrow \mathtt{chocolate}$, $\mathtt{Sol}(\mathtt{srce})$ solves the query $\mathtt{srce}$: $\mathtt{Sol}(\mathtt{srce})$ is such that $\mathtt{Sol}(\mathtt{srce}) \models_{\mathcal{O}} \mathtt{srce}$.

3. *Adaptation.* $\mathtt{AKB}$ is assumed to be empty, so to construct the adaptation path $\mathtt{AP}$, the substitution $\mathtt{chocolate} \rightsquigarrow \mathtt{baking\_chocolate}$ is generated automatically from σ_1. This substitution is further specialized into the substitution

$\Sigma_1 = \mathtt{cocoa} \rightsquigarrow \mathtt{baking\_chocolate}$, according to the axiom $\mathtt{cocoa} \Rightarrow \mathtt{chocolate}$ of $\mathcal{O}$. A first solution $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$ is computed by applying to $\mathtt{Sol}(\mathtt{srce})$ the adaptation path $\mathtt{AP} = \Sigma_1$. The user suggests that an ingredient is missing in $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$ but could not identify a repair strategy. An AK discovery is triggered in order to suggest gradual refinements of $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$.

4. *Choosing the Training Set.* The training set $\mathtt{TS}$ is chosen from Σ_1: AK is learned by comparing the recipes containing cocoa with the recipes containing baking chocolate. $\mathtt{TS}$ is composed of the set of variations $\Delta_{k\ell} \in \mathcal{L}_\Delta$ between pairs of recipes $(\mathtt{R}_k, \mathtt{R}_\ell) \in \mathtt{CB} \times \mathtt{CB}$ such that $\{\mathtt{cocoa}^-, \mathtt{baking\_chocolate}^+\} \subseteq \Delta_{k\ell}$.
5. *Mining and Filtering.* A value is given to the support threshold σ_s and the mining step outputs a set of variations. A filter retains only the variations that correspond to substitutions applicable to modify $\mathtt{Sol}(\mathtt{srce})$.
6. *Solution Test and Validation.* The user selects the learned variation $\Delta = \{\mathtt{cocoa}^-, \mathtt{baking\_chocolate}^+, \mathtt{oil}^-\}$ from the result set. Δ is interpreted as the substitution $\Sigma = \mathtt{cocoa} \land \mathtt{oil} \rightsquigarrow \mathtt{baking\_chocolate}$, which suggests to replace cocoa by baking chocolate in the retrieved recipe and to remove oil. The user explains this rule by the fact that baking chocolate contains more fat than cocoa, and therefore substituting cocoa by baking chocolate implies to reduce the quantity of fat in the recipe.

   Further solution refinements are proposed to the user. The set of learned variations is filtered in order to retain only the substitutions Δ' that are more specific than Δ, i.e., such that $\Delta \subseteq \Delta'$. Among the retained variations is the variation $\Delta' = \{\mathtt{cocoa}^-, \mathtt{baking\_chocolate}^+, \mathtt{oil}^-, \mathtt{vanilla}^-\}$, which is interpreted as the substitution $\Sigma' = \mathtt{cocoa} \land \mathtt{oil} \land \mathtt{vanilla} \rightsquigarrow \mathtt{baking\_chocolate}$. Σ' suggests to also remove vanilla in the recipe *Ultralight Chocolate Cake*. The user is satisfied with the refined solution $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$ resulting from the application of the adaptation path $\mathtt{AP} = \Sigma'$ to $\mathtt{Sol}(\mathtt{srce})$, so the reformulation $(\mathtt{baking\_chocolate} \rightsquigarrow \mathtt{chocolate}, \mathtt{cocoa} \land \mathtt{oil} \land \mathtt{vanilla} \rightsquigarrow \mathtt{baking\_chocolate})$ is added to the adaptation knowledge base $\mathtt{AKB}$.

5.5 A Second Example: Cooking a Chinese Soup

A second example is presented in which the AK discovery process is triggered in response to the user feedback in order to repair the adaptation presented in Sect. 3. In this example, the user is encouraged to formulate the cause of the adaptation failure. A repair strategy is chosen that is used to parametrize the AK discovery process.

1. *Representing the Target Problem.* In this example, the target problem $\mathtt{tgt}$ is:

$$\mathtt{tgt} = \mathtt{chinese} \land \mathtt{soup} \land \mathtt{leek} \land \neg\mathtt{peanut\_oil}$$

   In the TAAABLE interface, the field “Ingredients I Want” is filled in with the token $\mathtt{leek}$, the field “Ingredients I Don’t Want” is filled in with the token $\mathtt{peanut\_oil}$ and the field “Types I Want” is filled in with the tokens $\mathtt{chinese}$ and $\mathtt{soup}$.
2. *Retrieval.* As in Sect. 3, two substitutions $\sigma_1 = \neg\mathtt{peanut\_oil} \rightsquigarrow \emptyset$ and $\sigma_2 = \mathtt{leek} \rightsquigarrow \mathtt{onions}$ are generated automatically from the ontology $\mathcal{O}$. The

similarity path $\texttt{SP} = \sigma_2 \circ \sigma_1$ is applied to $\texttt{tgt}$ in order to produce the modified query $\texttt{srce} = \texttt{chinese} \wedge \texttt{soup} \wedge \texttt{onions}$. The system retrieves the recipe *Wonton Soup*, whose representation $\texttt{Sol(srce)}$ solves the query $\texttt{srce}$: $\texttt{Sol(srce)}$ is such that $\texttt{Sol(srce)} \vDash_O \texttt{srce}$.

3. *Adaptation.* Initially, $\texttt{AKB} = \emptyset$, so to construct the adaptation path $\texttt{AP}$, two substitutions $\Sigma_1 = \emptyset \rightsquigarrow \neg\texttt{peanut\_oil}$ and $\Sigma_2 = \texttt{green\_onion} \rightsquigarrow \texttt{leek}$ are automatically generated from σ_1 and σ_2.
4. *Solution Test and Validation.* The solution $\widetilde{\texttt{Sol}}\texttt{(tgt)}$ is presented to the user for validation, together with the adaptation path $\texttt{AP} = \Sigma_1 \circ \Sigma_2$ that was used to generate it.
5. *The User is Unsatisfied!* The user complains that the adapted recipe is practically unfeasible because the proposed solution $\widetilde{\texttt{Sol}}\texttt{(tgt)}$ does not contain oil anymore, and oil is needed to saute the bok choy.
6. *What has Caused the Adaptation Failure?* The cause of the adaptation failure is identified through interactions with the user. The user validates the intermediate solution $\widetilde{\texttt{Sol}}\texttt{(pb)}$ that results from the application of the substitution $\Sigma_2 = \texttt{green\_onion} \rightsquigarrow \texttt{leek}$ to $\texttt{Sol(srce)}$. But the user invalidates the solution $\widetilde{\texttt{Sol}}\texttt{(tgt)}$ that results from the application of $\Sigma_1 = \emptyset \rightsquigarrow \neg\texttt{peanut\_oil}$ to $\widetilde{\texttt{Sol}}\texttt{(pb)}$. The substitution Σ_1 is identified as responsible for the adaptation failure since its application results in the removal of oil in the recipe.
7. *Choosing a Repair Strategy.* A repair strategy is chosen according to the user's feedback. The user expresses the need for oil in the adapted recipe, so the repair strategy consists in replacing peanut oil by another oil. An AK discovery process is triggered to decide which oil to replace peanut oil with.
8. *Choosing the Training Set.* A set of recipes that contain peanut oil is compared with a set of recipes containing other types of oil. The training set $\texttt{TS}$ is composed of the set of variations $\Delta_{k\ell} \in \mathcal{L}_\Delta$ between pairs of recipes $(\texttt{R}_k, \texttt{R}_\ell) \in \texttt{CB} \times \texttt{CB}$ such that $\{\texttt{oil}^=, \texttt{peanut\_oil}^-\} \subseteq \Delta_{k\ell}$.
9. *Mining and Filtering.* A value is given to the support threshold σ_s and the mining step outputs a set of variations. A filter retains only the variations that correspond to substitutions applicable to modify $\texttt{Sol(pb)}$.
10. *Solution Test and Validation.* The user selects the learned variation $\Delta = \{\texttt{oil}^=, \texttt{peanut\_oil}^-, \texttt{olive\_oil}^+\}$ from the result set. Δ is interpreted as the substitution $\Sigma = \texttt{peanut\_oil} \rightsquigarrow \texttt{olive\_oil}$, which suggests to replace peanut oil by olive oil in the retrieved recipe. The adaptation path $\texttt{AP} = \Sigma \circ \Sigma_2$ is computed and the repaired solution $\widetilde{\texttt{Sol}}\texttt{(tgt)}$ is presented to the user for validation. The user is satisfied with the corrected solution $\widetilde{\texttt{Sol}}\texttt{(tgt)}$, so the reformulation $(\emptyset \rightsquigarrow \neg\texttt{peanut\_oil}, \texttt{peanut\_oil} \rightsquigarrow \texttt{olive\_oil})$ is added to the adaptation knowledge base $\texttt{AKB}$.

6 Discussion and Related Work

AK acquisition is a difficult task that is recognized to be a major bottleneck for CBR system designers due to the high knowledge-engineering costs it generates. To overcome these knowledge-engineering costs, a few approaches (e.g., [5,9,11]) have applied machine learning techniques to learn AK offline from differences between cases of the case base. In [11], a set of pairs of source cases is selected from the case base and each selected pair of source cases is considered as a specific adaptation rule. The featural differences between problems constitute the antecedent part of the rule and the featural differences between solutions constitute the consequent part. Michalski's closing interval rule algorithm is then applied to generalize adaptation rule antecedents. In [9], adaptation knowledge takes the form of a set of adaptation cases. Each adaptation case associates an adaptation action to a representation of the differences between the two source problems. Machine learning algorithms like C4.5 or RISE are applied to learn generalized adaptation knowledge from these adaptation cases in order to improve the system's case-based adaptation procedure.

When applying machine learning techniques to learn adaptation knowledge from differences between cases, one main challenge concerns the choice of the training set: which cases are worth comparing? Arguing that (1) the size of the training set should be reduced to minimize the cost of the adaptation rule generation process and that (2) the source cases that are worth comparing should be the ones that are more similar, only the pairs of source cases that were judged to be similar according to a given similarity measure are selected in [9] and [11]. However, committing to a particular similarity measure might be somewhat arbitrary. Therefore, in [5], the authors decided to include in the training set all the pairs of distinct source cases of the case base. This paper introduces a third approach: the choice of the training set is determined interactively and according to the problem-solving context, taking advantage of the fact that the AK discovery process is triggered on-line. This approach appears to be very promising since the learning algorithm can be parametrized in order to learn only the knowledge that is needed to solve the target problem.

The examples presented above also show that knowledge discovery techniques allow to come up with more complex adaptation strategies than the simple one-to-one ingredient substitutions generated from the ontology $\mathcal{O}$. In particular, these techniques can help identifying interactions between the different ingredients that appear in the recipes (like e.g., that cocoa contains less fat than baking chocolate, so oil should be removed) as well as co-occurrences of ingredients (like say, that cinnamon is well-suited with apples). Besides, adaptation knowledge is learned at different levels of generality, so the user can be guided into gradual solution refinements.

Several CBR systems make use of interactive and/or opportunistic knowledge acquisition approaches to improve their learning capabilities. For example, in Creek, an approach that combines case-based and model-based methods, general knowledge is acquired through interactions with the user [1]. This knowledge acquisition process is provided in addition to the traditional case acquisition and allows the system to acquire knowledge that cannot be captured through cases only. In the Dial system, adaptation knowledge is acquired in the form of adaptation cases: when a case has to be adapted, the adaptation process is memorized in the form of a case and can be reused to adapt another case. Hence, adaptation knowledge is acquired through a CBR process inside the main CBR cycle. It must be remarked that adaptation cases can either be built automatically by adaptation of previous adaptation cases or manually by a user who interactively builds the adaptation case in response to a problem by selecting the appropriates operations to perform [12]. Hence, knowledge acquisition in Dial appears to be both interactive and opportunistic. Chef is obviously related to the work described here [10]. Chef is a case-based planner in the cooking domain, its task is to build recipes on the basis of a user's request. The input of the system is a set of goals (tastes, textures, ingredients, types of dishes) and the output is a plan for a single recipe that satisfies all the goals. To solve this task, Chef is able to build new plans from old ones stored in memory. The system is provided with the ability to choose plans on the basis of the problems that they solve as well as the goals they satisfy, but it is also able to predict problems and to modify plans to avoid failures (plans are indexed in memory by the problems they avoid). Hence, Chef learns by providing causal explanations of failures thus marking elements as ”predictive” of failures. In other words, the acquired knowledge allows the system to avoid identical failures to occur again. In our approach, we propose to go one step further by using failure to acquire knowledge that can be more widely used.

7 Conclusion and Future Work

In this paper, a novel approach for adaptation knowledge acquisition is presented in which the knowledge learned at problem-solving time by knowledge discovery techniques is directly reused for problem-solving. An application is proposed in the context of the cooking CBR system TAAABLE and the feasibility of the approach is demonstrated on some use cases. Future work will include developing a graphical user interface and doing more extensive testing. Opportunistic and interactive knowledge discovery in TAAABLE implies that the user plays the role of the domain expert, which raises several issues. For example, how to be sure that the knowledge expressed by a particular user is valuable? How to ensure that the adaptation knowledge base will remain consistent with time? Besides, TAAABLE is meant to be multi-user, so if the system's knowledge evolves with experience, some synchronization problems might occur. Therefore, the envisioned multi-user, ever-learning TAAABLE system needs to be thought of as a collaborative tool in which knowledge acquired by some users can be revised by others.

References

1. Aamodt, A.: Knowledge-Intensive Case-Based Reasoning in Creek. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS, vol. 3155, pp. 1–15. Springer, Heidelberg (2004)
2. Badra, F., Lieber, J.: Representing Case Variations for Learning General and Specific Adaptation Rules. In: Cesta, A., Fakotakis, N. (eds.) Proceedings of the Fourth Starting AI Researcher's Symposium (STAIRS 2008), pp. 1–11 (2008)
3. Badra, F., Bendaoud, R., Bentebibel, R., Champin, P.-A., Cojan, J., Cordier, A., Després, S., Jean-Daubias, S., Lieber, J., Meilender, T., Mille, A., Nauer, E., Napoli, A., Toussaint, Y.: Taaable: Text Mining, Ontology Engineering, and Hierarchical Classification for Textual Case-Based Cooking. In: Schaaf, M. (ed.) Computer Cooking Contest - Workshop at European Conference on Case-Based Reasoning (ECCBR 2008), pp. 219–228 (2008)
4. Schaaf, M. (ed.) ECCBR Workshops, ECCBR 2008, The 9th European Conference on Case-Based Reasoning, Workshop Proceedings (2008)

5. d'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., Szathmary, L.: Case Base Mining for Adaptation Knowledge Acquisition. In: Proceedings of the International Conference on Artificial Intelligence, IJCAI 2007, pp. 750–756 (2007)
6. de Mántaras, R.L., Plaza, E.: Case-Based Reasoning: An Overview. AI Communications 10(1), 21–29 (1997)
7. Cordier, A.: Interactive and Opportunistic Knowledge Acquisition in Case-Based Reasoning, Phd Thesis, Université Lyon 1 (2008)
8. Cordier, A., Fuchs, B., Lana de Carvalho, L., Lieber, J., Mille, A.: Opportunistic Acquisition of Adaptation Knowledge and Cases - The IakA Approach. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS, vol. 5239, pp. 150–164. Springer, Heidelberg (2008)
9. Craw, S., Wiratunga, N., Rowe, R.: Learning Adaptation Knowledge to Improve Case-Based Reasoning. Artificial Intelligence 170(16-17), 1175–1192 (2006)
10. Hammond, K.: CHEF: A model of case-based planning. In: Proceedings of the 5th National Conference on Artificial Intelligence, pp. 267–271. AAAI Press, Menlo Park (1986)
11. Hanney, K., Keane, M.T.: The Adaptation Knowledge Bottleneck: How to Unblock it By Learning From Cases. In: Proceedings of the 2nd International Conference on CBR, pp. 359–370 (1997)
12. Leake, D., Kinley, A., Wilson, D.: Acquiring Case Adaptation Knowledge: A Hybrid Approach. In: Proc. of the 13th National Conference on Artificial Intelligence, pp. 684–689 (1996)
13. Melis, E., Lieber, J., Napoli, A.: Reformulation in Case-Based Reasoning. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 172–183. Springer, Heidelberg (1998)