Innovation awareness in case-based reasoning systems

Josep Lluís Arcos

IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia, Spain.
arcos@iiia.csic.es, http://www.iiia.csic.es

Abstract. The development of case-based reasoning systems that have to operate for a long time stresses the problem of domain evolving environments. In this paper we will present a procedure for dealing with real-world domains where case solutions are evolving along the time. We will exemplify the use of our approach in a deployed engineering design system.

1 Introduction

The development of case-based reasoning systems that have to operate for a long time stresses the problem of domain evolving environments. Because we are dealing with long-lived systems, the chances of changes on the domain environments increase. In this context, *sustained* case-based reasoning systems [1] are strongly needed.

Two main evolving directions can be identified. The first one is the change on the type of problems: new types of problems may become important and previously important problems may become irrelevant. The second one is the change on the type of solutions: the same type of problems that were previously solved using a specific domain solution may require a new domain solution to be solved.

Examples of case-based reasoning tasks that have to deal with evolving solutions are design and configuration tasks. Real-world design systems have to incorporate functionalities for dealing with the use of new design components or the improvement of previously existing components. we call this design problem the *innovation problem*.

An option for solving the innovation problem is to incorporate maintenance processes into the case-based reasoning applications [9, 6, 8]. The most usual techniques used for catching up the changes in the domain environment are case maintenance techniques for reorganizing the case base. Nevertheless, the success of any CBR system depends on all its knowledge containers [7] and, specially, on the similarity or retrieval knowledge and on the adaptation knowledge.

In this paper we will present a proposal for dealing with problem domains with evolving solutions concentrating the efforts on the retrieval and the reuse

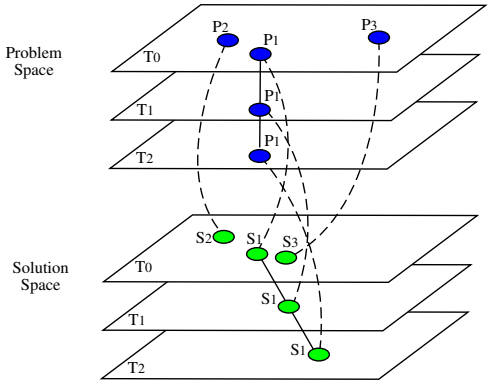

Fig. 1. An Extension of the classical problem and solution CBR spaces for dealing with time factor.

steps. Our goal is to develop an *innovative aware* CBR procedure, i.e. a CBR procedure where the retrieval and reuse phases are not degraded by the periodical incorporation of innovations in the problem solutions.

We are currently implementing a solution for the innovation problem in *T-Air*, a deployed engineering design system. *T-Air* is a case-based reasoning application developed for aiding engineers in the design of gas treatment plants [3].

This paper is organized as follows: In section 2 we present the problem and propose the innovation aware CBR procedure. In section 3 we exemplify the incorporation of the proposed procedure in a deployed application. The paper ends with a description of the current status of the work and the planned future work.

2 The innovation aware problem

The goal of a sustained CBR system is to not degrade the quality of the solutions generated taking into account that the domain environment may evolve. A first naive strategy for solving this problem can be to only use recently solved cases (for instance, including the case date to the similarity). Nevertheless, as we will see below this strategy is not enough.

In a given case base, we can classify the cases into two categories: *customary problems* and *occasional problems*. Customary problems are cases that are periodically solved by the system. For customary problems, the strategy of only focusing on recent solutions can be appropriate. The main problem rises with occasional problems. When a new occasional target problem has to be solved, usually only old solutions can be found in the case base. Then, a CBR inference system that is just reusing old solutions may generate solutions with low quality, i.e. solutions that may cause the distrust in the system.

Let us illustrate the evolving environment problem using the scheme of the Figure 1. There is a customary problem P_1 that was solved at times T_0, T_1, T_2. Each time the problem was solved, a small innovation was applied to the solution. Whenever a new P_1 target problem has to be solved, it is clear that we can take, as a basis for the reuse, the most recent solution stored in the case-base.

Now let us assume that there is an occasional problem P_2 that was solved at T_0 (see Figure 1), when a new P_2 target problem arises at T_2, we have two alternative cases to consider: S_2 solved at T_0 or S_1 solved at T_2. Using as a criterion the problem similarity, S_2 is the solution designed for the closest problem. Nevertheless, because problems P_1 and P_2 are very similar and P_2 has been solved recently, it can be more feasible to consider S_1 as a candidate for reuse (i.e using the case date when assessing similarity). Taking the last alternative, we are imposing a more powerful adaptation mechanism.

Finally, in the figure, we have another occasional problem P_3 that was solved at T_0. P_3 is a problem far from the other problems P_1 and P_2 but that has a solution S_3 close to the other solutions. Whether a new P_3 target problem comes to the system, P_1 and P_2 will not be retrieved. Thus, the closest solution is S_3. Nevertheless, taking into account that S_3 was designed at T_0, the solution we can reuse from S_3 will possibly not include recent innovations. Then, taking into account similar solutions more recently i.e. solved (solution S_1 that has been solved at T_2) we can improve the quality of the solution by tuning the solution taking into account the innovations introduced in S_1.

From these simple examples, it is clear that the innovation problem has to be dealt in the CBR inference procedure. Below we will present a variant of the classical CBR inference cycle [2] that incorporates an additional retrieval and a reuse step on the space of solutions.

Before describing our innovation aware procedure for dealing with evolving solutions, we will introduce some basic notation: a case c_i is defined as a tripled $c_i = (p_i, s_i, t)$ where p_i is the problem description, s_i is the solution, and t is the date when c_i was solved. Moreover, we assume that exists a similarity measure $Sim_p(p_i, p_j)$ between problem descriptions for retrieving and ranking the cases more similar to a new target problem. We say that two cases c_i, c_j are *innovation variants* when their problems p_i, p_j are equivalent ($Sim_p(p_i, p_j) = 1$) and their solutions s_i, s_j are different. Finally, we also assume that exists a similarity measure $Sim_s(s_i, s_j)$ between case solutions.

2.1 The innovation aware procedure

We propose to incorporate an additional retrieval and adaptation step into the usual CBR inference cycle: after retrieving and reusing the most similar cases for generating the solution of a new target problem, when the reused cases are not recent, we propose to refine the solution generated by looking for recentness paths on the solution space. Our CBR inference procedure is then divided into four phases:

1. Retrieving similar cases using Sim_p: given a new problem p_i, the first phase uses the Sim_p similarity measure on problem descriptions for retrieving and

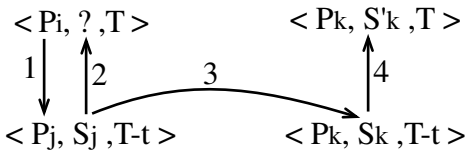

Fig. 2. A simplified diagram of the innovation aware procedure.

ranking similar cases (noted C). This phase models the usual retrieval step. Because we are not taking into account the solution date, all the cases with an equivalent problem description will be grouped with the same similarity.

2. First solution proposal: when the most similar cases are recent, their solutions are directly reused and next phases are skipped. Otherwise, a first solution s_i for p_i is constructed by reusing solutions of previously retrieved cases.
3. Retrieving similar solutions using Sim_s: the goal of the third phase is to retrieve, for each case $c_j = (p_j, s_j, t)$ in C, the cases S with a similar solution (using Sim_s) solved near t (plus or minus a fixed threshold δ). Moreover, only the solutions that have other cases in the case base that are innovation variants are considered.
4. Applying innovation variants: the goal of the last phase is to apply innovation variants to s_i. This phase requires domain specific policies for identifying the relevant solution changes.

Figure 2 shows a summary of the innovation aware procedure: first a retrieval on the problem space; then a reuse on solutions; next a retrieval on the solution space; and finally a reuse on the innovation path.

3 The application in an industrial system

We are currently incorporating the innovation aware CBR procedure in *T-Air*, a case-based reasoning application developed for aiding engineers in the design of gas treatment plants [3]. The gas treatment is required in many and diverse industrial processes such as the control of the atmospheric pollution due to corrosive residual gases which contain vapours, mists, and dusts of industrial origin. Examples of gas treatments are the absorption of gases and vapours such as SO_2, CLH, or CL_2; the absorption of NO_x with recovering of HNO_3; the absorption of drops and fogs such as PO_4H_3 or $ClNH_4$; dust removal in metallic oxides; and elimination of odours from organic origin.

The main problem in designing gas treatment plants is that the diversity of possible problems is as high as the diversity of industrial processes but there are only experimental models for few of them. The knowledge acquired by engineers with their practical experience is the main tool used for solving new problems.

T-Air uses a highly structured representation of cases. A case is represented as a complex structure embodying four different kinds of knowledge: the *Input Knowledge*, a *Chemical Case-Model*, the solution *Flow Sheet*, and *Annotations*.

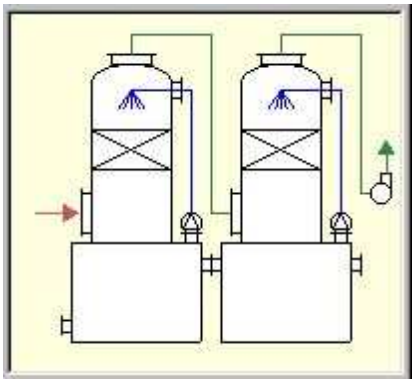

Fig. 3. An example of a simple flow sheet generated by *T-Air* with two scrubbers, each of them on top of a tank with a pump that sucks the washing liquid from the tank to the scrubber, and one fan at the end of the process.

The *Input Knowledge* embodies data about the customer such as the industrial sector it belongs or the industrial process that originates the polluting gas; data about the working conditions of the installation such as temperature or gas flow; data about the composition and concentration of input polluted gas; and data about the desired concentration in the output emission. The *Chemical Case-Model* embodies a graph structure, generated by *T-Air* using the chemical knowledge, modeling the main characteristics of the input gas. The chemical case-model extends the input knowledge and fixes the thresholds for the working conditions, the main issues to be analyzed, and the kind of gas treatment required. The *Flow Sheet* describes the solution designed for cleaning a polluted gas. As shown in Figure 3, the flow sheet specifies a collection of required equipment (mainly scrubbers, pumps, tanks, and fans), the design and working parameters for each equipment, and the topology of the installation (the gas circuit and the liquid circuits). The flow sheet is also represented as a graph structure. Finally, *Annotations* are meta-information that, when an external factor influences the solution, describe the reasons for a given solution decision—examples of single annotations are the design decisions forced by the user requirements such as the washing liquid, over-dimensionated parameters because of security reasons, or spatial requirements.

The *T-Air* inference process has been implemented using *constructive adaptation* [5], a generative technique for reuse in CBR systems. The design of a solution in *T-Air* is organized in four task levels: a) selecting the class of chemical process to be realized; b) selecting the major equipments to be used—and their inter-connections; c) assessing the values for the parameters of each equipment; and d) adding auxiliary equipment. Task levels a) and b) are mainly related with retrieval mechanisms. Tasks levels c) and d) are mainly related with adaptation mechanisms. A solution in *T-Air* is constructed by combining and adapting several previously solved designs. We use the input knowledge and

chemical knowledge (stored in the chemical case-models) as the basis for determining the similarity between a new problem and those treated previously and stored in the case base. Model-Based Reasoning is used in the adaptation stage for assessing the working parameters of each equipment of the flow-sheet.

The innovation problem in *T-Air* is due to the continuous improvements on the equipment used in the design of gas treatment plants. Specifically, the innovation in the scrubbers (the gas washing elements) that are the core elements in a gas treatment plant. It is not usual to incorporate new models of scrubbers. The usual procedure is to apply innovations to the current models. For instance, an innovation on the scrubber cover can decrease the pressure drop– i.e. increase the washing efficiency. Moreover, because there are many washing parameters estimated experimentally, the behavior and the in-site measurements of the deployed gas treatment plants is also a source of knowledge continuously incorporated into future designs.

When designing gas treatment plants two classes of problems can be identified: customary designs and occasional designs. Customary designs—for instance the gas treatment inside wastewater treatment plants—are good examples for tracking the innovations introduced in scrubbers. Solutions for occasional designs have to be tuned with customary designs. Otherwise, the quality of a solution for an occasional design may become very low.

The innovation aware procedure is only relevant for *T-Air* tasks levels b) and c): the selection of the major equipments to be used and the assessment of the values for the parameters of each equipment.

For assessing the similarity among case solutions (Sim_s), we have used *perspectives* [4]. Perspectives is a mechanism developed to describe declarative biases for case retrieval in structured and complex representations of cases. This mechanism is also very powerful for assessing similarities among case solutions where, as in design tasks, solutions are also represented as complex and structured representations.

The *T-Air* procedure for tasks b) and c) is being developed as follows:

1. Retrieving similar cases using Sim_p: *T-Air* retrieves cases based on the input requirements and the chemical case-model. Sim_p is mainly based on chemical knowledge that is used for determining the similarity between a gas composition in a new problem and those treated previously and stored in the case base.
2. First solution proposal: a new solution is generated by *each* "core equipment" found in the retrieved cases. For instance, odour elimination with absorption can be realized with the core equipment "two scrubbers, a pump, and a fan" or "one multiventuri, a pump, and a fan". Since the there are no analytical models capable of estimating the major parameters of scrubbers, the TECNIUM company experience (represented as cases) is the mainly available knowledge[1].

[1] There are less critical parameters that can computed using analytical methods, and *T-Air* uses them when available.

3. Retrieving similar solutions using Sim_s: the goal in this step is to retrieve customary designs with solutions close to the new solution proposals generated in the previous step.
4. Applying innovation variants: using the customary designs, the solution proposals generated in the second step are revised. This revision is performed by analyzing the evolution of the major parameters of scrubbers along the innovation variants. We are implementing a collection of equations and a collection of safety conditions expressed as heuristics for assessing the influence of the innovation variants in the target problem.

We are currently finishing the implementation of the last step. We are testing alternative equations and safety conditions.

4 Conclusions

We presented a CBR procedure for dealing with problem domains with evolving solutions concentrating the efforts on the retrieval and the reuse steps. The procedure is being developed for design and configuration tasks, where a solution has a complex and structured representation and is usually constructed with the contribution of different cases. The *innovative aware* CBR procedure manages the periodically incorporation of innovations in the problem solutions without decreasing the system performance. An additional retrieval and adaptation step is incorporated into the usual CBR inference cycle for capturing the innovations performed in customary problems and applying them to the solutions of occasional problems.

We are currently finishing the implementation of the innovation aware procedure in *T-Air*, a deployed application for aiding engineers in the design of gas treatment plants. The system performance with the test examples we are using for the development are encouraging. Nevertheless, we have not yet tested the system in a systematic way.

Acknowledgements

The author thanks Ramom López de Mántaras and Enric Plaza for their helpful and stimulating suggestions in the elaboration of this paper. The research reported in this paper was supported by the *T-Air* project financed by the company TECNIUM. Any possible mistake or error in the description of the working principles of chemical gas treatment is the sole responsibility of the author.

References

1. Agnar Aamodt. Knowledge-intensive case-based reasoning and sustained learning. In Luigia Aiello, editor, *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 1–6. Pitman Publishing, 1990.

2. Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39–59, 1994.
3. Josep Lluís Arcos. T-air: A case-based reasoning system for designing chemical absorption plants. In David W. Aha and Ian Watson, editors, *Case-Based Reasoning Research and Development*, number 2080 in Lecture Notes in Artificial Intelligence, pages 576–588. Springer-Verlag, 2001.
4. Josep Lluís Arcos and Ramon López de Mántaras. Perspectives: a declarative bias mechanism for case retrieval. In David Leake and Enric Plaza, editors, *Case-Based Reasoning. Research and Development*, number 1266 in Lecture Notes in Artificial Intelligence, pages 279–290. Springer-Verlag, 1997.
5. Enric Plaza and Josep Ll. Arcos. Constructive adaptation. In Susan Craw and Alun Preece, editors, *Advances in Case-Based Reasoning*, number 2416 in Lecture Notes in Artificial Intelligence, pages 306–320. Springer-Verlag, 2002.
6. Thomas Reinartz, Ioannis Iglezakis, and Thomas Roth-Berghofer. Review and restore for case-based maintenance. *Computational Intelligence*, 17(2):214–234, 2001.
7. Michael M. Richter. Introduction. In M. Lenz, B. Bartsch-Spörl, H.D. Burkhard, and S. Wess, editors, *CBR Technology: From Foundations to Applications*, pages 1–15. Springer-Verlag, 1998.
8. Barry Smyth and Elizabeth McKenna. Competence models and the maintenance problem. *Computational Intelligence*, 17(2):235–249, 2001.
9. David C. Wilson and David B. Leake. Maintaining case-based reasoners: Dimensions and directions. *Computational Intelligence*, 17(2):196–213, 2001.