A General Strategy for Adaptation in Case-Based Reasoning

Béatrice Fuchs[1], Jean Lieber[2], Alain Mille[1], Amedeo Napoli[2]
[1] LIRIS, Nautibus, 8 bd Niels Bohr, 69 100 Villeurbanne,
bfuchs@liris.univ-lyon1.fr, amille@liris.univ-lyon1.fr
[2] Orpailleur research group, LORIA, (CNRS, INRIA, Nancy Universities),
B.P. 239 54 506 Vandœuvre-lès-Nancy,
lieber@loria.fr, napoli@loria.fr.

Abstract

The case-based reasoning process relies on three main steps, retrieval, adaptation and learning. This article proposes a general, domain-independent, and operational formalization of the adaptation process, where a solution of the target problem is designed on the basis of the relations existing between a source case –problem and solution– and the target problem. The ideas underlying the article are that adaptation in case-based reasoning may be guided by a general strategy, relying on the decomposition of the target problem into (target) sub-problems, on the existence of local operators able to solve the sub-problems, and finally on the existence of global operators able to combine the set of local solutions for building a global solution of the target problem. This general formalization is used for defining a general strategy of adaptation that is operational and that is illustrated by a real-world application. This general strategy is of first importance, allowing a better understanding of the adaptation process, and a reuse of general adaptation operators for different real-world situations. In addition, this article aims at providing a general framework for studying the adaptation process, both from theoretical and practical points of view.

1 Introduction

The case-based reasoning process relies on three main steps, retrieval, adaptation and learning [36, 25]. This article holds on the second step of this process and proposes a general formalization of the adaptation step, based on the relations existing between a source problem and a target problem. This general formalization is of first importance, whereas in the majority of the situations, the adaptation process closely depends on the application being studied. This paper presents the results of studies carried on by the authors for several years on the formalization of the adaptation step within the CBR process. It extends and completes a first tentative of formalization that has been proposed in [13].

Let us suppose that are given a target problem and a case base including a collection of source cases, where a source case is a pair composed of a source problem and the associated source solution. During the retrieval step, several source problems of the case base are retrieved and compared to the target problem. The matching process establishes locally the correspondence between a feature –or a descriptor– of the source problem and the corresponding feature of the target problem. The similarities and differences between the features of the local source and target problem are made explicit. Then, a global similarity measure may combine the local feature similarities for assessing a global similarity between the source

and the target problem. Accordingly, a general schema of the adaptation process is proposed in figure 1, where a source case (on the left of the figure) has been selected by the retrieval process as the "best" candidate for solving the target problem, meaning that the solution of this best candidate can be reused for designing the solution of the target problem. The framework proposed for formalizing the adaptation process relies on two dimensions: (i) the vertical dimension corresponds to the case dimension, including the problem and the solution, with the source case on the left and the target case on the right, (ii) the horizontal dimension corresponds to the matching of source and target problems, and to the corresponding modification of the source solution for designing the target solution.

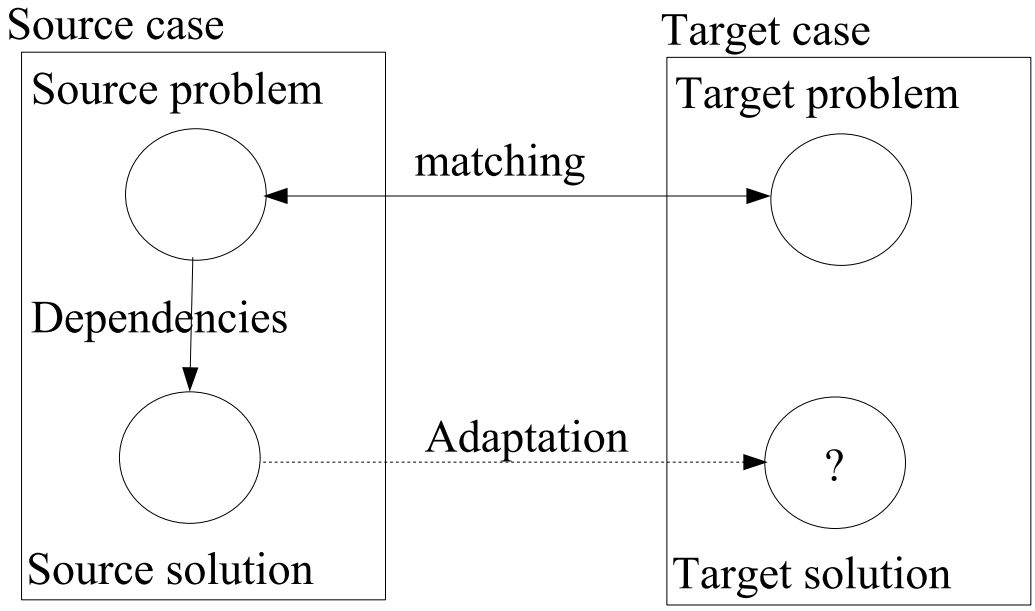

Figure 1: The general scheme of the adaptation process.

This is the task of the adaptation process to rely on the selected source case for designing a solution of the target problem. Firstly, the solution of the source case may be copied as a first solution of the target problem. Furthermore, this first and raw solution is modified for erasing the differences existing between the source and target problems, that have been pointed out by the matching process. The adaptation process is then guided both by the matching process, indicating what adjustments are needed, and by the *dependencies* indicating what are the problem features having an *influence* on the solution features. Given a difference between a source problem feature and the corresponding target problem feature, and a dependency between the source problem feature and a source solution feature, an elementary adaptation operation is in charge of modifying the source solution feature for designing the target solution feature.

As illustration, let us consider the simple following example, inspired from a talk of Ian Watson at ICCBR 99: knowing that $4 \times 4 = 16$, what is the solution of 4×5? The adaptation problem is the following: let the source problem be 4×4 whose solution is 16 (source case = source problem and source solution), what is the solution of the target problem 4×5? Here, the problem has two main features, namely the two operands, and the solution has one feature, namely the result. The differences between the features of the source and the target problems are 0 for the first operator and +1 for the second feature. The dependency between the source problem and the source solution may be expressed as follows: "the effect of an elementary variation of an operand on the result is proportional to the value of the

other operand". Consequently, considering that 4×5 is equal to $4 \times (4 + 1)$, the variation of the result caused by the variation of the operands in the target problem will be of 0 for the first operand (namely 4) and of $1 \times 4 = 4$ for the second operand (namely 5). The adapted solution feature will be given by the source solution feature and the effect of the variation using the + operator, i.e. $4 \times 5 = 16 + 0 + 4 = 20$.

Although this example is easy to understand and rather easy to solve, it shows all the elements that are involved within the adaptation process. Moreover, even if target problems are more complex in the real-world case, the adaptation process relies on the principles that are illustrated in the above example, namely a source problem, a source solution, dependencies and influences between features.

In this article, we promote the idea that adaptation in case-based reasoning may be guided by a general level strategy. This strategy relies first on the decomposition of the target problem into sub-problems, second on the existence and the use of local operators able to solve the sub-problems, third on a process able to combine the set of local solutions for building a global solution for the target problem. The decomposition of the target problem into sub-problems (simpler to solve) is achieved in the problem space, and performed according to the so-called similarity paths reifying a possible sequence of modification relations leading from the target problem to a source problem (or, from the target problem to the source problem). A modification relation characterizes an elementary modification, i.e. a local operator, that may be applied on either the target problem or the source problem. The similarity path has a corresponding path in the solution space, that allows to modify the source solution for building a target solution. The local operators in the problem space for splitting the target problem into sub-problems, have corresponding local operators in the solution space, for adapting the source solution into a target solution. A set of differences emphasizes the global difference between the source and target problems, while a set of dependencies emphasizes the relations between operations in the space of problems and the corresponding operations in the space of solutions.

The above principles are materialized within a practical and general algorithm for adaptation, that may be instantiated in an application domain for a specific purpose. Accordingly, the main contributions of the paper can be read as follows:

- A general strategy for taking into account the adaptation step in the case-based reasoning process exists and may be implemented according to domain-independent principles.
- This strategy is based on the general problem-solving method consisting in decomposing a complex problem into sub-problems more easy to be solved.
- The decomposition of the problem guides the decomposition of the adaptation process: to a local sub-problem corresponds a local solution, and the local solutions may be combined for building a global solution.

The metaphor that has been chosen for illustrating the different dependencies between the problem and its solution, and between the source and target problems, relies on (partial) differentials that materialize a dependence, knowing however that there is not an effective use of the differential calculus as in mathematics. Here, the differential materializes a dependence.

The framework for adaptation presented in this article is general and unifying. Indeed, the research work holding on adaptation usually relies on the fact that there exists a set of local operators that can be combined for building a global solution for the target problem. Most of the time, these local operators are domain-dependent and there is an *ad hoc* combination of the local solution for building a global solution. By contrast, the present strategy is

general and may be reused in different contexts or application domains, considering either local operators for problem decomposition or for solution adaptation.

The summary of the article is as follows. First, a general review of the state of the art on the adaptation process is proposed, with a historical perspective. Then, a framework for formalizing the adaptation process is proposed, relying on the two-dimensional point of view introduced in figure 1, i.e. the vertical case dimension and the horizontal matching/modification dimension. Based on these two dimensions, the relations between the problem and its solution are studied, and a general dependency-driven adaptation strategy is proposed, using a set of generic adaptation operators. The description of a real-world experience, namely the Prolabo experiment, illustrates the way this general adaptation strategy may be implemented. A discussion and a comparison with related work terminate the article.

2 Survey on adaptation in CBR

2.1 The pioneers of CBR and the adaptation process

Roger C. Schank [35, 26] and Janet Kolodner [27] probably have invented the expression "Case-Based Reasoning" for denoting the general process of reusing past experiences for facing new situations [36]. The CBR process is mainly memory-oriented and a model of dynamic memory is proposed for indexing and retrieving the best adapted memorized scripts, that fit a new situation and that only need a few adjustments to be reused. It is not so surprising that these first efforts for building a theory of the adaptation process were oriented towards memory search strategies, for finding structures or values which could be good candidates for replacing structures or values in the past experience to fit a new situation [17]. During the DARPA conference in 1989 [18], Janet Kolodner explained that the best case for a CBR system is the case that is the most useful to complete the reasoning process [24]. According to her definition –she was working on plan reuse– the best case is the case satisfying as well as possible the current goal and the easiest one to be adapted. The notion of "easiest case to be adapted" is of central concern within a CBR system. Nevertheless, Janet Kolodner only proposed generic similarity measures showing to which extent the current goal could be satisfied given the past experience.

2.2 Strategies for adaptation

Thomas Hinrichs considered two families of adaption strategies [21]: (1) *value selection* and (2) *structure modification*. The value selection relies on either a "local search principle" or a "transformation principle". The local search principle consists in navigating an available concept hierarchy for generalizing the value to replace, and then to specialize this value again, choosing a value satisfying as well as possible the constraints expressed by the target problem and violated by the source case. The transformation principle assumes that values are structured objects which can be transformed in new ones by adding, deleting or substituting parts of the objects. This point of view on adaptation enlightens the nature of adaptation: adaptation needs specific knowledge on the *dependencies between problem parts and solution parts* of a case. Moreover, these dependencies have to be taken into account within the similarity measure used to select a source case for solving a target problem, providing in this way a "semantics" to the similarity measure. The structure modification principle consists in merging equivalent information pieces in the same variable or, conversely, in splitting information in two different variables. In both cases, there is a simplification process transforming the source solution for making feasible the adaptation process. These principles are applied

for numerical values representing geometrical dimensions in [22].

2.3 The transformational and derivational approaches

Concurrently, a transformational approach for plan reuse was studied by J.G. Carbonell around 1983 [10]. A plan is compared to another one on the basis of the initial state, the final state, the respective constraints within the path described in the target problem and in the source problem, and the proportion of operator preconditions satisfied in the target problem situation. This measure is called the "applicability measure of a source solution" because it maximizes adaptation possibilities. Adaptation consists in solving a problem –of different nature called "transposed problem"– starting from an initial state, i.e. the source problem, to a final state, i.e. the target problem, through a set of intermediate states, where each one solves a difference pointed out at matching time. The intermediate states may correspond as well to meaningless plans. Thus, the adaptation process consists in building a "transposed plan space", i.e. a kind of parallel plan space of adapted plans corresponding to the intermediate states of the planification problems (as introduced above), giving birth to an adaptation path. A set of adaptation operators is proposed for solving each step of the adaptation path (11 so-called T-operators are valid in the transposed space). This way of representing an adaptation path can be connected to the so-called "reformulations" and "similarity paths" presented in [31, 29] and discussed hereafter. Following the same idea, in [32], it is proposed to build an adaptation from the source case by exploring successively (best-first search) hypotheses allowing to get closer to the desired goal (configuration domain problems).

J.G. Carbonell developed later another point of view characterized by a "derivational approach" also based on analogical reasoning [11]. He noticed that it is possible to keep a trace of the use of the different transformation operators used in a plan. Consequently, instead of switching from the source plan space to the transposed plan space, he proposed to "replay" operator selection rules by substituting the source context for the target context in the rules. J.G. Carbonell stresses the fact that the applicability measure is modified since problems are considered as similar when the initial reasoning process is similar. The reasoning process has to be adapted to the new context, and then to be replayed within in it. An example is given concerning the adaptation of a sort function in Pascal to a sort function in LISP. Actually, the "case" to be adapted is the reasoning process itself, using a transformational approach. The complexity of this plan adaptation approach has been studied in [3]. Plan reuse and reasoning process adaptation have inspired numerous theoretical works, as in [19], where the authors consider adaptability as the measure of the adaptation effort in terms of the corresponding computation complexity. In [23], this theoretical work is completed, and it is demonstrated that the adaptation effort cannot be easily controlled, and that this approach needs knowledge for solving the problem from scratch. A nice survey of case-based planning may be found in [37].

2.4 Adaptation categories

In 1993, a survey of various adaptation processes, either transformational or derivational, has been reported in the book of Janet Kolodner [25]:

- Copy of the source solution.
- Modification of the source solution,
  - by substitution:

    - with re-instantiation, i.e. abstraction and specialization,
    - with parameter adjustment, i.e. dependency with respect to a problem item,
    - with memory search for a set of possible close values:
        - using local search, i.e. selecting an existing value close to the source value,
        - using a specialized search, i.e. a heuristic for selecting a value beyond close ones;
  - by transformation:
    - using "common sense" transformations,
    - using a repair process guided by a domain model;
  - by plan derivation replay:
    - searching methods or explanations in the source case,
    - applying again concerned methods in the target case.

This list makes more precise the proposals of Thomas Hinrichs from 1989. Parameter adjustment was reported in different articles of the DARPA 89 Conference, without being completely explained in terms of knowledge units to be reused either for adaptation or for similar case retrieval.

Another taxonomy of adaptation knowledge units was reported in [20], according to the role played in the adaptation process at the different steps of the CBR cycle: operators for the elaboration of the target problem, operators for role substitution, operators for subgoals decomposition, and operators for dependence management. This taxonomy was not initially aimed at qualifying working knowledge for adaptation, but appeared to be very useful to highlight it.

2.5 Adaptation as constraint solving

Case-based design is one of the major application domain of CBR [30]. Several works were published on adaptation formalization involving a constraint-solving approach: [22] introduces the notion of "surface case descriptors" that does not include complete knowledge for reasoning (thus excluding a derivational approach). An important improvement of the operators of Thomas Hinrichs is proposed within the JULIA system [21]: numerical constraints are expressed on geometric patterns instead of on symbolic terms.

Some authors consider adaptation as a set of source case modifications which are constrained by the target problem. There exists a set of integrity constraints represented by a set of equalities or inequalities to be satisfied for solving the target problem. Based on a source case satisfying at best the target constraints, just a few dimensions have to be taken into account for adaptation. The general adaptation methodology is the following:

1. Select a case satisfying mandatory constraints: this step is usually performed by the user.
2. Select an initial set of parameters covering the target constraints: this dimension expansion" step is usually performed by the user too.
3. Establish whether the system is over-constrained (too many constraints and thus no solution), under-constrained (not enough constraints and thus too many solutions), or has just one solution. This "dimension analysis step" may be automatized.

4. When the system is over-constrained, the selection of an initial set of parameters (step 2) may be replayed.

5. If the system is under-constrained, apply a "dimension reduction" process.

6. Solve the residual constraints with a classical constraint solver.

This approach points out the notion of "influence" in the selection of a source case, by minimizing the set of violated constraints to satisfy the target problem, for minimizing in turn the adaptation cost.

2.6 CBR "without" adaptation

CBR is widely used for classification or interpretation tasks that consist in assigning a target problem to a specific class, with respect to its similarity with a well-classified source problem [28]. A source case or "source class" may be seen an abstraction of solutions of similar problems.

Here, the interesting step within the CBR process is the retrieval step and the associated retrieval knowledge: the adaptation step is not necessary and the problem is solved as soon as a source class has been found, providing a generic solution for the target problem. Knowledge units for similarity may be acquired according to three approaches: (1) the inductive approach is based on automatic learning from data [2], using the information contribution of case descriptors for selecting the source class, (2) "ad hoc" approaches building similarity or dissimilarity measures according to more or less general methods [34]; (3) an "explanation-driven" approach using a few well chosen source problems for explaining a source class [5]. This last approach relies on the explicit building of dependency relationships between problem descriptors and an abstract solution class. This approach stresses the differences between the different classes, and applies quite well to problem-solving since it provides necessary information for adaptation [1, 6, 15, 14].

3 A general strategy of adaptation

3.1 Basic notations

(** articuler **)

A case is the association of a problem and of a solution to this problem: $(\texttt{pb}, \texttt{Sol(pb)})$. A source case is denoted by $(\texttt{srce}, \texttt{Sol(srce)})$, where `srce` is the *source problem* and `Sol(srce)` is the *source solution*. A target case is denoted by $(\texttt{tgt}, \texttt{Sol(tgt)})$, where `tgt` is the *target problem* and `Sol(tgt)` is the *target solution* that has to be built by the adaptation process.

Cases (problems and solutions) are represented by sets of *descriptors*. A descriptor d is defined by a pair $d = (a, v)$ where a is an attribute and v is the value associated to this attribute. The following convention is adopted hereafter: lower cases are used for problem descriptors –$d = (a, v)$– and upper cases, for solution descriptors—$D = (A, V)$. In accordance with this vocabulary, source and target cases are defined as follows:

- $\texttt{srce} = \{d_1^0, \ldots d_n^0\} = \{d_i^0\}_{i=1\ldots n}$, where $d_i^0 = (a_i^0, v_i^0)$ is a descriptor of the source problem;
- $\texttt{Sol(srce)} = \{D_1^0, \ldots D_N^0\} = \{D_j^0\}_{j=1\ldots N}$ where $D_j^0 = (A_j^0, V_j^0)$ is a descriptor of the source solution;

- $\texttt{tgt} = \{d_1^q, \ldots d_n^q\} = \{d_i^q\}_{i=1\ldots n}$ where $d_i^q = (a_i^q, v_i^q)$ is a descriptor of the target problem;
- $\texttt{Sol(tgt)} = \{D_1^q, \ldots D_N^q\} = \{D_j^q\}_{j=1\ldots N}$ where $D_j^q = (A_j^q, V_j^q)$ is a descriptor of the target solution.

Let us consider a simple example of the same type as the example introduced in section 1. Let 4×4 be the source problem, whose solution is 16, and 3×5 be the target problem. Conforming to this formalism, the adaptation specification is described as follows:

$$\begin{aligned}
\texttt{srce} = \{&d_1^0 = (a_1^0 = \texttt{1}^{\texttt{st}}\ \texttt{operand}, v_1^0 = 4),\\
&d_2^0 = (a_2^0 = \texttt{2}^{\texttt{nd}}\ \texttt{operand}, v_2^0 = 4)\}\\
\texttt{tgt} = \{&d_1^q = (a_1^q = \texttt{1}^{\texttt{st}}\ \texttt{operand}, v_1^q = 3),\\
&d_2^q = (a_2^q = \texttt{2}^{\texttt{nd}}\ \texttt{operand}, v_2^q = 5)\}\\
\texttt{Sol(srce)} = \{&D_1^0 = (A_1^0 = \texttt{result}, V_1^0 = 16)\}\\
\texttt{Sol(tgt)} = \{&D_1^q = (A_1^q = \texttt{result}, V_1^q = \ ?)\}
\end{aligned}$$

3.2 From source to target

In this section, we are interested in the "horizontal view" of figure 1, i.e., on the relationships between the source and the target. This horizontal view of adaptation can be decomposed in two main steps:

Similarity assessment (or *matching process*) is the study of the relationships between `srce` and `tgt`. It aims at pointing out what makes `srce` and `tgt` *similar* and what makes them *dissimilar*. The similarity assessment provides a *matching* from `srce` to `tgt` denoted by $\mathcal{M}(\texttt{srce}, \texttt{tgt})$. The profile of the similarity assessment, which specifies its inputs and output, is

$$(\texttt{srce}, \texttt{tgt}) \mapsto \mathcal{M}(\texttt{srce}, \texttt{tgt})$$

Solution modification is the study of the relationships between $\texttt{Sol(srce)}$ and (what will be) $\texttt{Sol(tgt)}$. This study is based on the matching $\mathcal{M}(\texttt{srce}, \texttt{tgt})$, thus its profile is

$$(\texttt{srce}, \texttt{Sol(srce)}, \texttt{tgt}, \mathcal{M}(\texttt{srce}, \texttt{tgt})) \mapsto \texttt{Sol(tgt)}$$

In the following, we are interested in two types of matchings (and to the corresponding solution modifications): the descriptor matching approach and the approach based on similarity paths. The former approach is based on the descriptors that compose the cases. The latter approach aims at decomposing the matching and the solution modification in several "simple" steps.

3.2.1 Descriptor matching

The descriptor matching approach is based on the elements composing problems, i.e., their descriptors: the question "How can two problems be matched?" is reduced to the question "What descriptors of two problems can be matched, and how?" The descriptor matching consists in matching descriptors having the same attribute names. This means that the similarity assessment associates to `srce` and `tgt` a set $\mathcal{M}(\texttt{srce}, \texttt{tgt})$ of triples $(d_i^0, d_i^q, \Delta d_i)$

such that $d_i^0 = (a_i, v_i^0) \in \texttt{srce}$ and $d_i^q = (a_i, v_i^q) \in \texttt{tgt}$ have the same attribute name $a_i = a_i^0 = a_i^q$. Therefore, a matching can be written

$$\mathcal{M}(\texttt{srce}, \texttt{tgt}) = \{(d_i^0, d_i^q, \Delta d_i)\}_i \tag{1}$$

where Δd_i encodes the *differences* between the values v_i^0 and v_i^q, i.e., some pieces of information about their similarities and dissimilarities. Δd_i is computed thanks to a domain-dependent operator denoted by $\ominus$:

$$\Delta d_i = v_i^q \ominus v_i^0$$

The solution modification following this similarity assessment is detailed in section 3.4.

3.2.2 Matching by similarity paths

We assume that `Problems`, the collection of any problems, is structured with a finite set of relations between problems denoted by $\mathcal{R}$. The space $(\texttt{Problems}, \mathcal{R})$ is called the *problem space*. The matching from `srce` to `tgt` is assumed to be a *similarity path*, i.e., a path in the problem space:

$$\mathcal{M}(\texttt{srce}, \texttt{tgt}) = \left(\texttt{pb}^0\, \texttt{r}^1\, \texttt{pb}^1\, \texttt{r}^2\, \texttt{pb}^2 \ldots \texttt{pb}^{q-1}\, \texttt{r}^q\, \texttt{pb}^q\right)$$
$$\text{with } \texttt{pb}^0 = \texttt{srce},\ \texttt{pb}^q = \texttt{tgt} \text{ and } \texttt{r}^i \in \mathcal{R}, i \in \{1, 2, \ldots q\}$$

The problems $\texttt{pb}^1$, $\texttt{pb}^2$, ... $\texttt{pb}^{q-1}$ are created by the matching process and are called *intermediate problems*.

The relations $\texttt{r} \in \mathcal{R}$ are assumed to have the following property:

$$\begin{array}{l} \textbf{if } \texttt{pb}\ \texttt{r}\ \texttt{pb}' \qquad (\texttt{pb} \text{ is related to } \texttt{pb}' \text{ by } \texttt{r}) \\ \quad \textbf{then} \text{ any solution } \texttt{Sol}(\texttt{pb}) \text{ of } \texttt{pb} \text{ can be} \\ \qquad \text{adapted into a solution } \texttt{Sol}(\texttt{pb}') \text{ of } \texttt{pb}'. \end{array} \tag{2}$$

This adaptation is performed thanks to a *specific adaptation function* with the following profile:

$$\mathcal{A}_\texttt{r} : (\texttt{pb}, \texttt{Sol}(\texttt{pb}), \texttt{pb}') \mapsto \texttt{Sol}(\texttt{pb}')$$
$$\text{for } \texttt{pb}, \texttt{pb}' \in \texttt{Problems}, \text{ such that } \texttt{pb}\ \texttt{r}\ \texttt{pb}'.$$

The ordered pair $(\texttt{r}, \mathcal{A}_\texttt{r})$ is called a *reformulation*. The set of the available reformulations constitute the available adaptation knowledge.

Once the similarity path is found, the solution modification can be processed simply by following this path in the solution space:

(1) The solution $\texttt{Sol}(\texttt{srce}) = \texttt{Sol}(\texttt{pb}^0)$ of $\texttt{srce} = \texttt{pb}^0$ is adapted into a solution $\texttt{Sol}(\texttt{pb}^1)$ of $\texttt{pb}^1$ thanks to $\mathcal{A}_{\texttt{r}^1}$;

(2) The solution $\texttt{Sol}(\texttt{pb}^1)$ of $\texttt{pb}^1$ is adapted into a solution $\texttt{Sol}(\texttt{pb}^2)$ of $\texttt{pb}^2$ thanks to $\mathcal{A}_{\texttt{r}^2}$;

...

(q) The solution $\texttt{Sol}(\texttt{pb}^{q-1})$ of $\texttt{pb}^{q-1}$ is adapted into a solution $\texttt{Sol}(\texttt{pb}^q)$ of $\texttt{pb}^q$ thanks to $\mathcal{A}_{\texttt{r}^q}$.

These solution modification steps can be performed because property (2) holds for each $(\texttt{r}^i, \mathcal{A}_{\texttt{r}^i})$: as soon as a similarity path is found, the solution modification process is ensured. Figure 2 shows such a solution modification.

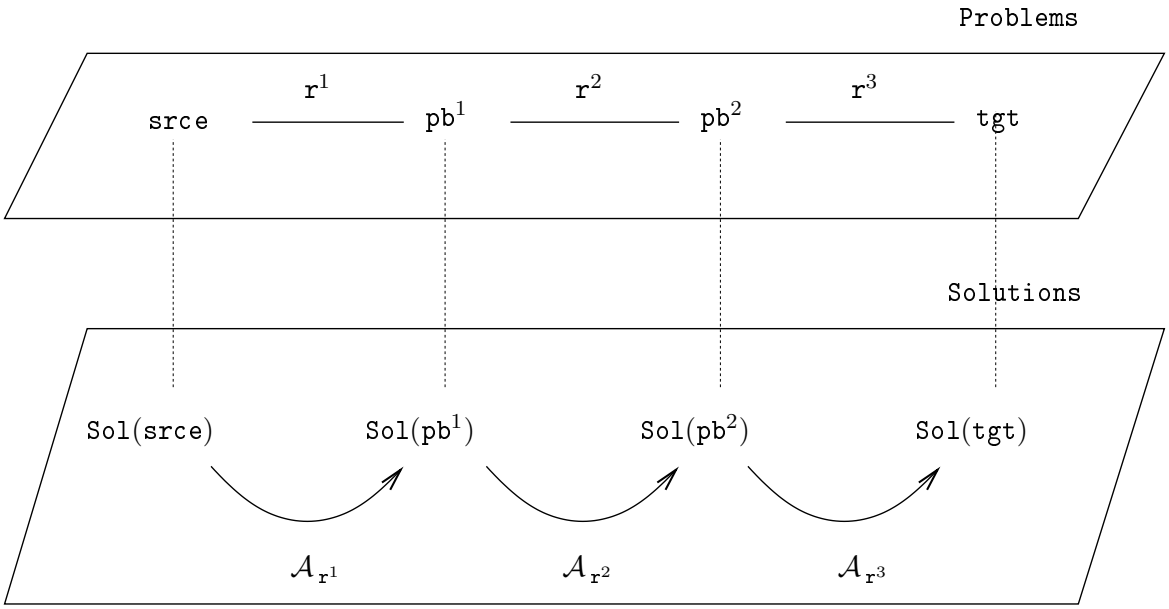

Figure 2: Solution modification following a similarity path.

Combining descriptor matching and matching by similarity path. The two previous approaches of matching can be combined as follows: each of the step of a similarity path, $\mathtt{pb}^k\,\mathtt{r}^k\,\mathtt{pb}^{k+1}$, can be represented by a descriptor matching $\mathcal{M}(\mathtt{pb}^k, \mathtt{pb}^{k+1})$:

$$\mathcal{M}(\mathtt{pb}^k, \mathtt{pb}^{k+1}) = \{(d_i^k, d_i^{k+1}, \Delta d_i^k)\}_i \quad (3)$$
$$\text{where } \Delta d_i^k = v_i^{k+1} \ominus v_i^k$$

Therefore, the similarity path matching between $\mathtt{srce}$ and $\mathtt{tgt}$ is composed of the $\mathcal{M}(\mathtt{pb}^k, \mathtt{pb}^{k+1})$:

$$\mathcal{M}(\mathtt{srce}, \mathtt{tgt}) = \{\mathcal{M}(\mathtt{pb}^k, \mathtt{pb}^{k+1})\}_k \quad (4)$$

On the previous simple example, a similarity path from $\mathtt{srce} = 4 \times 4$ to $\mathtt{tgt} = 3 \times 5$ can be defined by introducing a single intermediate problem, $\mathtt{pb}^1 = 3 \times 4$ as shown in figure 3. In this example, the differences Δd_i^k are computed by a numerical difference:

$$\Delta d_i^k = v_i^{k+1} - v_i^k$$

The *abstract operator* $\ominus$ is instantiated by the numerical difference between real numbers: $\ominus = -$. For example, $\Delta d_1^0 = v_1^1 - v_1^0 = 3 - 4 = -1$.

$$\underset{4\times 4}{\mathtt{srce}} \xrightarrow[\Delta d_2^0 = 0]{\Delta d_1^0 = -1} \underset{3\times 4}{\mathtt{pb}^1} \xrightarrow[\Delta d_2^1 = +1]{\Delta d_1^1 = 0} \underset{3\times 5}{\mathtt{tgt}}$$

Figure 3: The similarity path linking the source problem 4×4 and the target problem 3×5. The problem $\mathtt{pb}^1 = 3 \times 4$ is an intermediate problem.

3.3 From problem to solution

In this section, we are interested in the "vertical view" of figure 1. We assume that there exists relations between the problem and its solution called *dependencies* indicating that some

problem descriptors influence some solution descriptors. A dependency expresses that the variation of a problem descriptor has an influence on the variation of a solution descriptor.

Let $\texttt{pb}^k$ be a problem and $\texttt{Sol}(\texttt{pb}^k)$ be a solution of $\texttt{pb}^k$ (if $k = 0$, $\texttt{pb}^k = \texttt{pb}^0 = \texttt{srce}$, if $k = q$, $\texttt{pb}^k = \texttt{pb}^q = \texttt{tgt}$, else $\texttt{pb}^k$ is an intermediate problem of a similarity path). Let $d_i^k \in \texttt{pb}^k$ and $D_j^k \in \texttt{Sol}(\texttt{pb}^k)$. If the variation of d_i^k have an influence on the variation of D_j^k, then the dependency of d_i^k on D_j^k is a triple $(d_i^k, D_j^k, \mathcal{I}(D_j^k/d_i^k))$. d_i^k is called the *influencing* and D_j^k is called the *influencee* of the dependency. $\mathcal{I}(D_j^k/d_i^k)$ is part of the adaptation knowledge that has to be modelled and consists in an *influence function* indicating the impact of the influencing on the influencee. The influence function is at the basis of the adaptation operators of the adaptation strategy presented in section 3.4.

The set of dependencies of $\texttt{pb}^k$ descriptors on $\texttt{Sol}(\texttt{pb}^k)$ descriptors is denoted by $\mathcal{D}(\texttt{pb}^k, \texttt{Sol}(\texttt{pb}^k))$:

$$\mathcal{D}(\texttt{pb}^k, \texttt{Sol}(\texttt{pb}^k)) = \{(d_i^k, D_j^k, \mathcal{I}(D_j^k/d_i^k))\}_{i,\,j}$$

In the example, for each $k \in \{0, 1\}$, there are two dependencies between descriptors of $\texttt{pb}^k$ and the descriptor of $\texttt{Sol}(\texttt{pb}^k)$. For one operand, the influence function expresses that the effect of the variation of this operand is proportional to the other operand:

$$\begin{aligned}
\mathcal{D}(\texttt{srce}, \texttt{Sol}(\texttt{srce})) = \mathcal{D}(\texttt{pb}^0, \texttt{Sol}(\texttt{pb}^0)) &= \{(d_1^0, D_1^0, \mathcal{I}(D_1^0/d_1^0)),\\
&\quad (d_2^0, D_1^0, \mathcal{I}(D_1^0/d_2^0))\}\\
\mathcal{I}(D_1^0/d_1^0) &= v_2^0 = 4\\
\mathcal{I}(D_1^0/d_2^0) &= v_1^0 = 4\\
\mathcal{D}(\texttt{pb}^1, \texttt{Sol}(\texttt{pb}^1)) &= \{(d_1^1, D_1^1, \mathcal{I}(D_1^1/d_1^1)),\\
&\quad (d_2^1, D_1^1, \mathcal{I}(D_1^1/d_2^1))\}\\
\mathcal{I}(D_1^1/d_1^1) &= v_2^1 = 4\\
\mathcal{I}(D_1^1/d_2^1) &= v_1^1 = 3
\end{aligned}$$

This is summarized in figure 4.

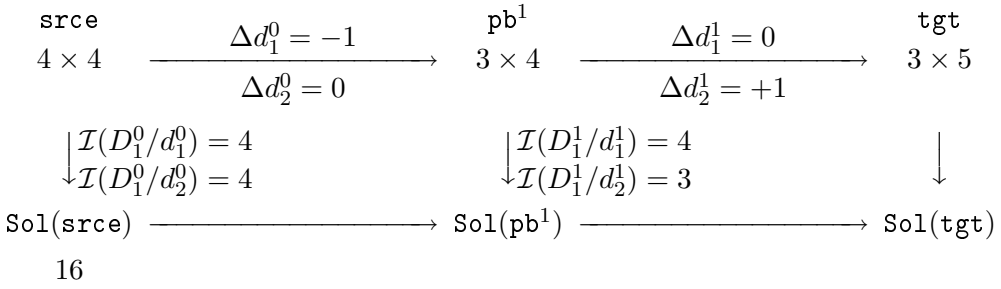

Figure 4: The influence functions between problems and solutions.

3.4 Generic adaptation operators

Classically, similarity assessment evaluates a similarity local to a descriptor and further a global similarity combines local similarities. In a similar way, we propose to assess the local influence of problem descriptors on solution descriptors, and then further to assess a global influence for each solution descriptor by combining local influences. The global influences are

finally mapped to the values of source solution descriptors in order to obtain the values of the target solution descriptors.

A parallel of these principles can be established with partial derivatives:

$$dy_j = \sum_i \frac{\partial y_j}{\partial x_i} \times dx_i$$

where dy_j features the variation on a given target solution descriptor which is obtained by combining several combinations of influence functions featured by $\frac{\partial y_j}{\partial x_i}$ and variations of problem descriptors featured by dx_i.

3.4.1 Local variation

The adaptation process combines together several elementary adaptation operations, each of them expressing the contribution of a given problem descriptor to a solution descriptor. An elementary adaptation operation denoted by $\Delta_i D_j^k$ is obtained thanks to a difference Δd_i^k and an influence function $\mathcal{I}(D_j^k/d_i^k)$ that are combined using an abstract operator $\otimes$:

$$\Delta_i D_j^k = \mathcal{I}(D_j^k/d_i^k) \otimes \Delta d_i^k$$

$\Delta_i D_j^k$ is the contribution of the variation of a problem descriptor d_i^k to the variation of a solution descriptor D_j^k and $\otimes$ is an abstract operator expressing how to combine the difference Δd_i^k between problem descriptors and the influence $\mathcal{I}(D_j^k/d_i^k)$ of this problem descriptor on the solution descriptor D_j^k.

This abstract operator $\otimes$ has to be instantiated in a given application domain. In the domain of our example it simply consists in a product between numbers: $\otimes = \times$.

$$\begin{cases} \Delta_1 D_1^0 = \mathcal{I}(D_1^0/d_1^0) \times \Delta d_1^0 = 4 \times (-1) = -4 \\ \Delta_2 D_1^0 = \mathcal{I}(D_1^0/d_2^0) \times \Delta d_2^0 = 4 \times 0 = 0 \end{cases}$$

$$\begin{cases} \Delta_1 D_1^1 = \mathcal{I}(D_1^1/d_1^1) \times \Delta d_1^1 = 4 \times 0 = 0 \\ \Delta_2 D_1^1 = \mathcal{I}(D_1^1/d_2^1) \times \Delta d_2^1 = 3 \times (+1) = +3 \end{cases}$$

3.4.2 Global variation

Once the individual contributions $\Delta_i D_j^k$ of each problem descriptor to a given solution descriptor has been assessed, they are gathered in a global contribution ΔD_j^k expressed using an abstract operator $\oplus$:

$$\Delta D_j^k = (\ldots(\Delta_1 D_j^k \oplus \Delta_2 D_j^k) \oplus \ldots \oplus \Delta_n D_j^k)$$

It is assumed that $\oplus$ is associative and commutative. Thus the expression above can be written:

$$\Delta D_j^k = \bigoplus_i \Delta_i D_j^k$$

ΔD_j^k is the global variation that has to be applied to D_j^k in order to obtain D_j^{k+1} by combining all the local variations $\Delta_i D_j^k$ of d_i^k on D_j^k.

In the example, $\oplus = +$:

$$\Delta D_1^0 = \Delta_1 D_1^0 + \Delta_2 D_1^0 = -4 + 0 = -4$$
$$\Delta D_1^1 = \Delta_1 D_1^1 + \Delta_2 D_1^1 = 0 + 3 = +3$$

3.4.3 Adaptation of target solution descriptors

For $k \in \{0, 1, \ldots, q-1\}$, in order to compute the value of the solution descriptor D_j^{k+1} of $\texttt{Sol}(\texttt{pb}^{k+1})$ from the solution descriptor D_j^k of $\texttt{Sol}(\texttt{pb}^k)$, the value of ΔD_j^k is used in the following way:

$$V_j^{k+1} = V_j^k \oplus \Delta D_j^k \tag{5}$$

Therefore, to compute the descriptors D_j^q of $\texttt{Sol}(\texttt{tgt})$ starting from the known descriptors D_j^0 of $\texttt{Sol}(\texttt{srce})$, the equation (5) is used with $k = 0$, $k = 1$, ..., and, finally $k = q - 1$.

In the example,

$$V_1^1 = V_1^0 \oplus \Delta D_1^0 = 16 + (-4) = 12$$
$$V_1^2 = V_1^1 \oplus \Delta D_1^1 = 12 + 3 = 15$$

Thus, $\texttt{Sol}(3 \times 5) = 15$.

The figure 5 summarizes the adaptation process.

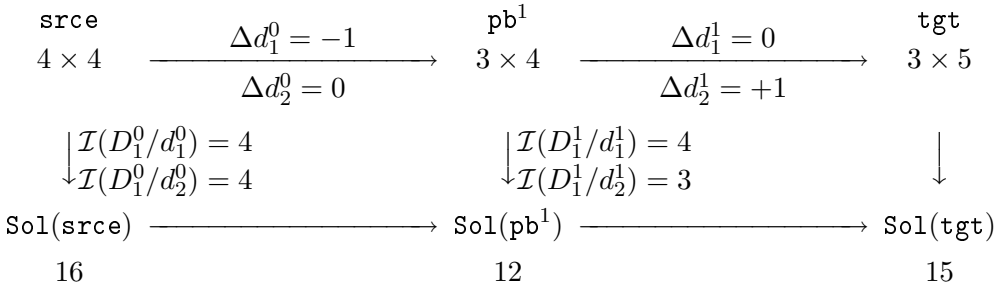

Figure 5: The sequence of adaptation operations from the source problem 4×4 to the target problem 3×5.

(** articulation**)

4 The Prolabo application

4.1 An overview of the "Prolabo" application

Prolabo is a company manufacturing and marketing products and devices for chemical, pharmaceutical, biochemical and biological laboratories. One of these devices is a guided microwave digester, i.e. a device aimed at preparing product samples for chemical analysis thanks to various analysis processes. The analysis process is performed by an analyzer, whose type depends on the digester type. The analysis process needs also that the samples are only constituted of the chemical atomic elements of the products to be analyzed. The guided microwave digester is in charge of breaking all molecular bonds between atoms using either special chemical agents, e.g. aggressive chemical agents, or microwave effects, e.g. mechanical and thermal effects. The guided microwave digester relies on injection pumps controlling the special chemical agent injection, and on a magnetron controlling the microwave effects. This

Descriptors	Type	Id
Analyzer Type	Symbol	d_1
Injection Speed	Real	d_3
Magnetron power	Real	d_4
Tube capacity	Real	d_5
Max power gradient	Real	d_6
Analysis class	Symbol	d_7
Sample weight	Real	d_8
Lipids Quantity	Real	d_9
Glucide Quantity	Real	d_{10}
Mineral Quantity	Real	d_{11}
Cellulose Quantity	Real	d_{12}
Water Quantity	Real	d_{13}

Table 1: The descriptors of a problem.

device is fully automated: injection pumps are driven according to three main parameters, namely the chemical agents to be injected, the injection speed, and the injection duration, while the main parameters of the magnetron are the power value and the emission duration.

A digestion program is composed of a number of sequential steps (from 5 to 20 steps) where each step is controlled by the five above parameters (actually, this list is a simplified one): choice of the chemical agent to be injected, injection speed of the chemical agent, injection duration, magnetron power (percent of the magnetron maximum power), and magnetron powering duration.

A case includes the description of a problem with its associated solution. The problem describes a generic digestion program, called hereafter a digestion plan, and the problem descriptors are: digestion constraints, analysis process constraints (see figures 9 and 10). In this paper, for confidentiality reasons and for the sake of simplicity, only a part of the descriptors is considered (see Table 1). The solution is composed of a synthesis of the digestion program (see figures 11 and 12) that can be processed further by an automate designing the associated digestion program (see for example figure 13). For a given target problem and a retrieved source case, adaptation is performed according to dependencies existing between source solution descriptors and source problem descriptors.

There exists a straightforward conceptual dependency between the analyzer type and the digestion plan type (see figure 6). The value of D_1 (Plan Type) depends on the value of d_1 (Analyzer Type) of the concerned analyzer. The more the analyzer is sensitive to aggressive chemical agents the more the digestion plan is moderated with respect to the energy provided within each step.

The dependencies between the numerical descriptors of the problem and the numerical descriptors of the solution are expressed by a ratio. For example, the value of D_{j3} (Total Energy) to be provided within a step of type j depends on d_8 (Sample Weight). The higher is the value of "Sample Weight", the higher is the value of the "Total Energy" to be provided within a step of type j.

It must be noticed that the dependency relations are only true for given "differences". When the "difference" between the source and the target problem descriptors is greater than a given threshold, the existing dependencies for building the corresponding solution descriptors cannot work anymore. Moreover, this kind of knowledge is local (see figures 7 and 8).

Hereafter, a concrete example is detailed, involving problem and solution descriptors.

Digestion program description		
Plan type	Symbol	D_1
Moderation level	Integer	D_2
Type by type step descriptions		
Type 1		
Number	Integer	D_{11}
Total duration	Real	D_{12}
Total energy	Real	D_{13}
Injected product	Symbol	D_{14}
Injected quantity	Real	D_{15}
...		
Type j		
Number	Integer	D_{j1}
Total duration	Real	D_{j2}
Total energy	Real	D_{j3}
Injected product	Symbol	D_{j4}
Injected quantity	Real	D_{j5}
...		

Table 2: The solution descriptors are the five digestion plan parameters.

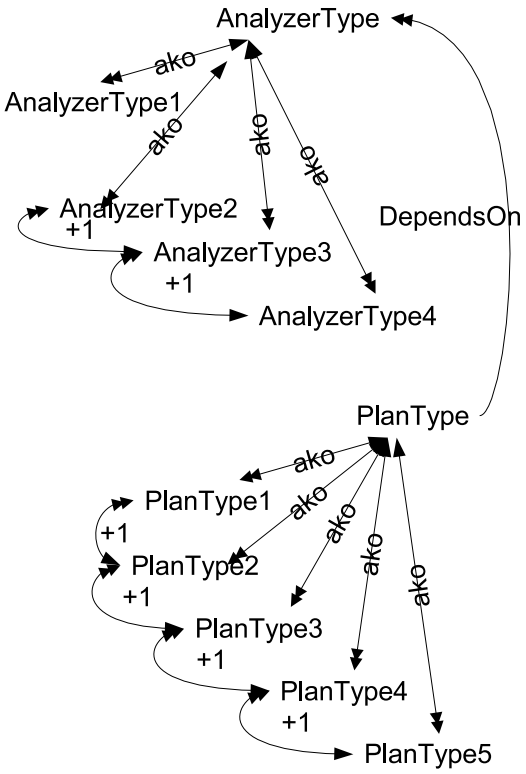

Figure 6: The dependencies between the analyzer type and the plan type.

4.2 A case study

4.2.1 A case description

For this example, we consider the following problem descriptors corresponding to the following attributes:

Problem attributes		
Identifier	Acronym	Definition
a_1^0	AT	Analyzer Type
a_7^0	AC	Analysis Class
a_8^0	SW	Sample Weight

The solution descriptors are the followings (there are only three different types of program steps, namely type 1, type 2 and type 3):

Solution attributes		
Identifier	Acronym	Definition
A_1^0	PT	Plan Type
A_2^0	ML	Moderation Level
A_{13}^0	TE1	Total Energy for steps of type 1
A_{14}^0	IP1	Injected Product for steps of type 1
A_{15}^0	IV1	Injected Volume for steps of type 1
A_{23}^0	TE2	Total Energy for steps of type 2
A_{24}^0	IP2	Injected Product for steps of type 2
A_{25}^0	IV2	Injected Volume for steps of type 2
A_{33}^0	TE3	Total Energy for steps of type 3
A_{34}^0	IP3	Injected Product for steps of type 3
A_{35}^0	IV3	Injected Volume for steps of type 3

The attributes of the target problem are the followings:

Target problem descriptors			
Identifier	Attribute	Value	Definition
d_1^q	AT	AnalyzerType1	Analyzer Type
d_7^q	AC	AnalysisClass3	Analysis Class
d_8^q	SW	0.8	Sample Weight

The attributes of the (retrieved) source case are the followings:

Source problem descriptors			
Identifier	Attribute	Value	Definition
d_1^0	AT	AnalyzerType3	Analyzer Type
d_7^0	AC	AnalysisClass6	Analysis Class
d_8^0	SW	0.6	Sample Weight

Source solution descriptors			
Identifier	Attribute	Value	Definition
D_1^0	PT	PlanType5	Plan Type
D_2^0	ML	4	Moderation Level
D_{13}^0	TE1	156	Total Energy for steps of type 1
D_{14}^0	IP1	Product3	Injected Product for steps of type 1
D_{15}^0	IV1	12	Injected Volume for steps of type 1
D_{23}^0	TE2	0	Total Energy for steps of type 2
D_{24}^0	IP2	Null	Injected Product for steps of type 2
D_{25}^0	IV2	0	Injected Volume for steps of type 2
D_{33}^0	TE3	120	Total Energy for steps of type 3
D_{34}^0	IP3	Product1	Injected Product for steps of type 3
D_{35}^0	IV3	18	Injected Volume for steps of type 3

These cases are processed as explained hereafter.

4.2.2 The description of influences

The influence $\mathcal{I}(D_1/d_1)$ of "Analyzer Type" acts on "Plan Type": the larger is the numerical label of the analyzer type, the more the plan type can be "blended", in taking care of not combining dangerous products. This is a conceptual influence stating that if the value of the source analyzer type is larger of one degree than the value of the target analyzer type, then one or more degrees have to be added to the value of the solution plan type (in the source case). Thus, descriptors such as "AnalyzerType" or "PlanType" are ordered according to discrete degrees, represented by an integer, e.g. AnalyzerType3, PlanType5...

The influence of D_1 on d_1 is computed by a function f_1, whose values are recorded in a table, returning the number of degrees to add or to subtract to the analyzer type. This function is not linear, and depends on the source value of "Analyzer Type" and on the magnitude of the difference Δd_i. The influence $\mathcal{I}(D_1/d_1)$ reads as follows:

$$\boxed{\mathcal{I}(D_1/d_1) = f_1(d_1, D_1, \Delta_{d_1})} \tag{6}$$

The influence $\mathcal{I}(D_2/d_7)$ of "Analysis Class" acts on "Moderation Level": the larger is the "Analysis Class", the more the analysis is constrained by the presence of volatile products, and, hence, the more the digestion program has to be "moderated", i.e. involving weaker temperature gradients in each step (entailing generally more steps in the program). A function f_2 is used to compute the Influence value $\mathcal{I}(D_2/d_7)$:

$$\boxed{\mathcal{I}(D_2/d_7) = f_2(d_7, D_2, \Delta_{d_7})} \tag{7}$$

The figure 8 illustrates how could be graphically represented the computation of Influence for symbolic values.

The influence $\mathcal{I}(D_{i5}/d_8)$ of "Sample Weight" acts on product quantity at each step type i. The larger is "Sample Weight", the larger is the product volume to be injected. The function f_3 used to compute this influence is not linear, and $\mathcal{I}(D_{i5}/d_8)$ has the following general form:

$$\boxed{\mathcal{I}(D_{i5}/d_8) = f_3(d_8, D_{i5}, \Delta_{d_8})} \tag{8}$$

The influence $\mathcal{I}(D_{i3}/d_8)$ of Sample Weight acts also on the "Total Energy" to be provided at each step of type i. The larger is the weight value the larger is the total energy value provided. The influence $\mathcal{I}(D_{i3}/d_8)$ reads as follows:

$$\boxed{\mathcal{I}(D_{i3}/d_8) = f_4(d_8, D_{i3}, \Delta_{d_8})} \tag{9}$$

Figure 7 illustrates the computation of an influence in the case of numerical values.

It can be noticed that function descriptions and abacus for the influence function are quite easy to obtain from domain experts, while it is impossible to find out the function formula for directly computing a target solution descriptor from a source solution descriptor (no theoretical and practical knowledge is available).

4.2.3 The adaptation of the source solution descriptors for building a target solution

The matching between `srce` and `tgt` is defined by:

$$\mathcal{M}(\texttt{srce}, \texttt{tgt}) = \{(d_i^0, d_i^q, \Delta d_i = v_i^q \ominus v_i^0)\}_i$$
$$\ominus : (x, y) \in \mathbb{Z}^2 \mapsto x \ominus y = x - y \in \mathbb{Z}, \text{ for } i \in \{1, 7\}$$
$$\Delta d_1 = v_1^q - v_1^0 = -2,$$
$$\Delta d_2 = v_7^q - v_7^0 = -3$$

$$\ominus : (x, y) \in \mathbb{R}^2 \mapsto x \ominus y = x - y \in \mathbb{R}, \text{ for } i = 8$$
$$\Delta d_8 = v_8^q - v_8^0 = 0.2,$$

$$\mathcal{M}(\texttt{srce}, \texttt{tgt}) = \{(d_1^0, d_1^q, \Delta d_1 = -2), (d_2^0, d_2^q, \Delta d_2 = -3), (d_8^0, d_8^q, \Delta d_8 = 0.2)\}$$

Dependencies between `srce` and `Sol(srce)` are given by:

$\mathcal{D}(\texttt{srce}, \texttt{Sol(srce)}) = \{(d_i^0, D_j^0, \mathcal{I}(D_i^0/d_j^0))\}$ with

$\mathcal{I}(D_1^0/d_1^0) = f_1(AnalyzerType3, PlanType5, -2) = 0$ for all i and j, $i \neq j$.

This means that for each type of analyzer, a type difference lower than 3 does not entail plan type modification (i.e. null influence).

$\mathcal{I}(D_7^0/d_2^0) = f_2(AnalysisClass6, 4, -3) = 1$ meaning that for this analysis type and this difference, a difference of one degree on the analysis type entails a difference of one degree on "Moderation Level".

$$\mathcal{I}(D_8^0/d_{15}^0) = f_3(0.6, 12, 0.2) = 25$$
$$\mathcal{I}(D_8^0/d_{25}^0) = f_3(0.6, 0, 0.2) = 0$$
$$\mathcal{I}(D_8^0/d_{35}^0) = f_3(0.6, 18, 0.2) = 25$$
$$\mathcal{I}(D_8^0/d_{13}^0) = f_4(0.6, 156, 0.2) = 200$$
$$\mathcal{I}(D_8^0/d_{23}^0) = f_4(0.6, 0, 0.2) = 0$$
$$\mathcal{I}(D_8^0/d_{13}^0) = f_4(0.6, 120, 0.2) = 200$$

A difference of 1 between problem descriptor values involves an influence that may be computed from the case base (taking into account non linearity if needed).

An elementary variation of a source descriptor value is computed by:

$$\Delta_i D_j = \Delta d_i \otimes \mathcal{I}(D_j^0/d_i^0)$$

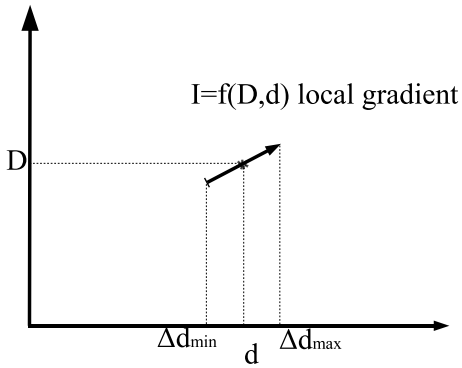

Figure 7: Numerical influence. In the numerical case, the influence $\mathcal{I}$ is the the same for any value in the interval Δ_d.

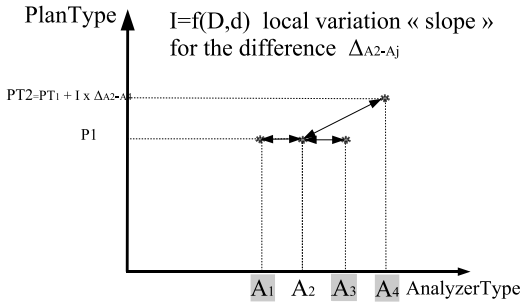

Figure 8: Symbolic influence. The difference between the analyzer type A2 to the analyzer type A3 yields a null influence on the plan type.

The $\otimes$ operator is defined as follows:

$$\otimes : (x, y) \in \mathbb{R}^2 \mapsto x \otimes y = x \times y \in \mathbb{R}$$

The elementary variations are the followings:

$$\Delta_1 D_1 = \Delta d_1 \otimes \mathcal{I}(D_1^0/d_1^0)$$
$$\Delta_1 D_1 = -2 \times 0 = 0$$

$$\Delta_7 D_2 = \Delta d_7 \otimes \mathcal{I}(D_2^0/d_7^0)$$
$$\Delta_7 D_2 = -3 \times 1 = -3$$

$$\Delta_8 D_{i5} = \Delta d_8 \otimes \mathcal{I}(D_{i5}^0/d_8^0) \text{ for i} \in \{1, 2, 3\}$$
$$\Delta_8 D_{15} = 0.2 \times 25 = 5$$
$$\Delta_8 D_{25} = 0.2 \times 0 = 0$$
$$\Delta_8 D_{35} = 0.2 \times 25 = 5$$

$$\Delta_8 D_{i3} = \Delta d_8 \otimes \mathcal{I}(D_{i3}^0/d_8^0) \text{ for i} \in \{1, 2, 3\}$$
$$\Delta_8 D_{13} = 0.2 \times 200 = 40$$
$$\Delta_8 D_{23} = 0.2 \times 0 = 0$$
$$\Delta_8 D_{33} = 0.2 \times 200 = 40$$

In the equation defining an elementary adaptation, i.e. $V_j^q = V_j^0 \oplus \Delta D_j$, the $\oplus$ operator is defined as follows:

$$\oplus : (x, y) \in \mathbb{R}^2 \mapsto x \oplus y = x + y \in \mathbb{R}$$

Recall that the target problem is described by the following descriptors:

$$\begin{array}{l} d_1^q = (AT, AnalyzerType1), \\ d_7^q = (AC, AnalysisClass3), \\ d_8^q = (SW, 0.8) \end{array}$$

The value of a target solution descriptor D_i^q is computed as stated above. For example, the descriptor $D_1^q = (PT, V_1^q) = (PT, V_1^0 \oplus \Delta D_1)$ is computed as $D_1^q = (PT, PlanType5 + 0) = (PT, PlanType5)$. The descriptor $D_{15}^q = (IV1, V_{15}^q) = (IV1, V_{15}^0 \oplus \Delta D_{15})$ is computed as $D_{15}^q = (IV1, 12 + 5) = (IV1, 17)$.

The full target solution `Sol(tgt)` reads as:

Target solution descriptors
$D_1^q = (PT, PlanType5 + 0) = (PT, PlanType5)$
$D_2^q = (ML, 4 + (-3)) = (ML, 1)$
$D_{13}^q = (TE1, 156 + 40) = (TE1, 196)$
$D_{14}^q = (IP1, Product3) = (IP1, Product3)$
$D_{15}^q = (IV1, 12 + 5) = (IV1, 17)$
$D_{23}^q = (TE2, 0 + 0) = (TE2, 0)$
$D_{24}^q = (IP2, null) = (IP2, null)$
$D_{25}^q = (IV2, 0 + 0) = (IV2, 0)$
$D_{33}^q = (TE3, 120 + 40) = (TE3, 160)$
$D_{34}^q = (IP3, Product1) = (IP3, Product1)$
$D_{35}^q = (TE3, 18 + 5) = (IV3, 23)$

The Prolabo application enlightens two important points on knowledge engineering for case-based reasoning:

1. Case elaboration needs a rather large amount of knowledge on the way of adapting a case. Actually, the (internal) representation of a case for being manipulated by the case-based reasoning process is probably different from the mental user representation. The Prolabo experience (and some other industry applications) shows that the representation of a case must be well-known by the current user of the case-based reasoning system. Accordingly, in the Prolabo application, a graphical synthesis of the case being processed within reasoning is proposed to the user (see figure 12).

2. Influences constitute the main knowledge units for adaptation, and therefore for similarity assessment as well. These kinds of influences do not generally depend on linear global functions, but rather on local functions depending on the value of the source solution descriptor, the value of the source problem descriptor, and on the magnitude of the difference to be adjusted between problem descriptors. In some application domains, knowledge on influences may be easily available from domain experts. Probably knowledge on influences may be elicited and/or mined (using an automatic learning process) from a set of cases put in correspondence by an expert.

5 Discussion and Related Work

The state of the art shows that unless adaptation is presented as the most important step in the case-based reasoning process taken into account by authors working on problem-solving applications, there is only a few research works on the formalization of adaptation.

In [6], there is a proposition of using domain knowledge for explaining the solution of a specific problem. According to this approach, domain knowledge provides knowledge units for similarity measures as adaptation operations as well. Actually, the paper highlights the key role played by dependencies between solution descriptors and problem descriptors. In the same way, a generic method is presented in [8, 7], dealing with generalization and specialization. The case generalization is carried out within a learning phase of the system, by organizing cases in an abstraction tree. The case specialization is carried out within a problem-solving phase, using a planner performing a heuristic search in the solution space. This approach relies both on a case model but also on a relevant model of the domain as needed by a planner such as STRIPS.

In [33], there is a proposition of a formalization of the adaptation in the context of design problems based on a particular case representation allowing case processing with CSP (constraint satisfaction problems) methods. Following a similar approach as in [8, 7], cases are split down into sub-cases, and global consistency is guaranteed by the constraint-solving method, that relies on an efficient heuristic: the source cases minimizing the constraints to be solved have to be preferred. "Consistency" here and hereafter means that the solution is actually a working solution of the target problem. In this context, dependencies are expressed as constraints to be satisfied, and the solver needs a detailed domain model for delivering an efficient processing.

The two preceding approaches are mainly related to generative case adaptation, while some research works have addressed transformational and substitution adaptation processes. In [9], adaptation knowledge is presented under the form of local functions transforming a source case into a target case according to expected quality measures. Adaptation is then performed by applying a set of adequate transformation functions allowing an improvement, i.e. a better quality measure, of the target case. Hence, there is a need for a global function allowing the composition of local quality improvements into a global quality improvement. In a certain way, this approach is rather close to the approach presented in this paper. Moreover, authors make the hypothesis that there exists a kind of "feed-back" associated with the solution in each stored case used for the measure of quality improvement. By default, each case is considered as having the best possible quality. Finally, it must be noticed that this approach does not guarantee a complete consistency of the adapted solution.

In [12], a simple local adaptation method uses interpolation functions for adapting a source solution descriptor depending on an observed difference between the source and target problem descriptors. Several interpolation techniques are enumerated according to the type of the descriptors: digital values, symbolic values, fuzzy quantifiers, ordinals, etc. This kind of interpolation requires a partial order relationship between descriptor values for working (especially between arbitrary symbolic values). This approach does not guarantee the consistency of the adapted solution, that must be in turn tested by another mean. By contrast, the knowledge units needed for guiding retrieval and adaptation are local, and may be more easily acquired by mining a case base or by eliciting expert knowledge (actually, a similar remark applies to the Prolabo experience, see 4).

In [4], the general idea "the most similar case is the case that is the easiest to adapt" is considered as the basic principle. Accordingly, the authors introduce a "metric" based on the relations existing between the problem descriptors and the solution descriptors, i.e. an

adaptation function for adapting a source solution into a target solution, provided that the matching between source and target is satisfied (a "maintenance" function, external to the adaptation process, is also available). Moreover, an "adaptation cost" guides the choice of the best source case among the cases satisfying the matching condition. A kind of "topology" can then be defined for selecting the most adaptable case during the retrieval step. This formal approach has not been implemented, and has not been extended, as far as we know.

In [16], the adaptation model is based on substitution as an adaptation operator. A case is connected to two individuals representing the problem and its solution in the considered case. Each solution individual is related through "dependency relations" to a set of individuals that are elementary descriptors of the solution. The substitution algorithm for adaptation propagates changes, i.e. substitutions on the descriptors of source solution, on solution descriptors as follows: (i) the list of the source solution elements to be adapted is built from the relations characterizing the observed differences between source and target problem descriptors, (ii) each element to be modified is substituted by a new one, taking into account the differences between the source and the target problems, and then the dependencies between solution descriptors. The search of substitutions to be performed relies on a specialized and guided search in an associated knowledge base.

6 Conclusion and perspectives

In this paper, we have presented a general and domain-independent formalization of the adaptation step within the CBR process. Several new and generic ideas have been introduced and discussed in the paper.

Firstly, adaptation is viewed as a central step in the CBR process for designing a solution of the target problem, based on the relations existing between a source case and a target problem. These relations are considered according to two main dimensions: (i) the vertical dimension refers to the case dimension and is based on the correspondence between the problem and its solution, (ii) the horizontal dimension refers to the matching of the source and target problems in the problem space, and to the corresponding modification/adaptation of the source solution for designing the target solution in the solution space. A general algorithm reifies these ideas and is detailed, taking into account these two dimensions for building a solution of the target problem.

Another important idea underlying the article is that adaptation is guided by a general strategy, relying on the decomposition of the target problem into sub-problems. Accordingly, local operators exist and have the ability to solve the sub-problems. Then, global operators control the local problem-solving processes and are able to merge the local solutions for building a global solution of the target problem.

In addition, this article aims at proposing a general framework for the adaptation process, i.e. a general strategy of adaptation that is operational and that can be used both for theoretical and practical purposes. This general strategy allows a better understanding of the adaptation process within the CBR process, and provides guidelines and general adaptation operators to be reused in real-world situations. Moreover, a complex real-world application is detailed and may be used as a referring example for other future practical applications.

Going further, the CBR process needs, in several aspects, to be guided by domain knowledge, at every step of the process. Domain knowledge may be used either by the human in charge of the system or by the system itself for designing a solution. It should become actual and important to consider CBR as a powerful inference schema, to be used for completing deduction and induction schemes in implemented knowledge-based systems, aimed at solving

real-world problems. In such a context, a complete system should take advantage of a case base and a knowledge base. Moreover, regarding the present needs, e.g. Semantic Web applications, such a complete system should be coupled with a KDD system –knowledge-discovery in databases– able to feed the case and knowledge bases. The design of a theoretical and practical framework for combining CBR, knowledge-based system technology, and knowledge-discovery technology, is an important next challenge, from the point of view of the authors, to be studied and made fully operational.

References

[1] Agnar Aamodt. Explanation-driven retrieval, reuse and learning of cases. In *EWCBR 93, Otzenhauzen, Germany*, pages 279–284, 1993.

[2] David Aha. Editorial. *Artificial Intelligence Review*, 11(1-5):1–6, 1997. Special Issue on Lazy Learning.

[3] Tsz-Chiu Au, Héctor Muñoz-Avila, and Dana Nau. On the complexity of plan adaptation by derivational analogy in a universal classical planning framework. In *6th European Conference, ECCBR-2002*, Aberdeen, Scotland, UK, 2002.

[4] Paolo Avesani and Enrico Blanzieri. Adaptation-dependent retrieval problem: A formal definition. In *Proceedings of ICCBR'99*, 1999.

[5] Ray Bareiss, Bruce Porter, and Creg Wier. Protos- an exemplar based learning apprentice. *International Journal of Man-Machines Studies*, 29:549–561, 1988.

[6] Ralph Bergmann, Gerd Pews, and Wolfgang Wilke. Explanation-based similarity: a unifying approach for integrating domain knowledge into case-based reasoning for diagnosis and planning tasks. In Stephan Wess, Klaus-Dieter Althoff, and Michael M. Richter, editors, *Workshop on Case-Based Reasoning*, Topics in Case-Based Reasoning, pages 182–196. Springer, Berlin, 1994.

[7] Ralph Bergmann and Wolfgang Wilke. Building and refining abstract planning cases by change of representation language. *Journal of Artificial Intelligence Research*, 3:53–118, 1995.

[8] Ralph Bergmann and Wolfgang Wilke. Paris : Flexible plan adaptation by abstraction and refinement. In A. Voß, R. Bergmann, and B. Bartsch-Spörl, editors, *Workshop on Adaptation in Case-Based Reasoning, ECAI-96*, Budapest, Hungary, August 1996.

[9] Ralph Bergmann and Wolfgang Wilke. Towards a new formal model of transformational adaptation in case-based reasoning. In Henri Prade, editor, *ECAI 98, 13th European Conference on Artificial Intelligence*, pages 53–57. John Wiley and Sons, Ltd, 1998.

[10] Jaime G. Carbonell. Learning by analogy: Formulating and generalizing plans from past experience. In R.S. Michalsky, J.G. Carbonnel, and T.M. Mitchel, editors, *Machine Learning: an Artificial Intelligence Approach*, pages 137–162. Tioga, Palo Alto, 1983.

[11] J.G. Carbonell. Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning, an Artificial Intelligence Approach, Volume II*, pages 371–392. Morgan Kaufmann Publishers, Inc., Los Altos, California, 1986.

[12] N. Chatterjee and J.A. Campbell. Interpolation as a means of fast adaptation in case-based problem solving. In Ralph Bergmann and Wolfgang Wilke, editors, *Fifth German Workshop on Case-Based Reasoning*, pages 65–74, Kaiserslautern, Germany, 1997.

[13] Béatrice Fuchs, Jean Lieber, Alain Mille, and Amedeo Napoli. An Algorithm for Adaptation in Case-based Reasoning. In W. Horn, editor, *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000), Berlin*, pages 45–49. IOS Press, Amsterdam, 2000.

[14] Béatrice Fuchs and Alain Mille. Explanation driven adaptation. In *proceedings of the workshop on adaptation in CBR*, 1996. Workshop on Case Based Reasoning - ECAI96.

[15] Béatrice Fuchs, Alain Mille, and Benoit Chiron. Operator decision aiding by adaptation of supervision strategies. *Lecture Notes in Artificial Intelligence vol 1010, First International Conference, ICCBR95, Sesimbra, Portugal*, pages 23–32, 1995.

[16] Pedro A. Gonzalez-Galero, Mercedes Gòmez-Albarràn, and Belén Diaz-Agudo. A substitution-based adaptation model. In *Proceedings of ICCBR'99*, 1999.

[17] Kristian J. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task.* Academic Press, Boston, 1989.

[18] Kristian J. Hammond, editor. *Workshop on case-based Reasoning, DARPA 89*, Pensacola Beach, Florida, 1989. Morgan-Kaufmann, San Mateo.

[19] Steve Hanks and Daniel S. Weld. A domain-independent algorithm for plan adaptation. *Journal of Artificial Intelligence Research*, 2:319–360, 1995.

[20] Kathleen Hanney, Mark T. Keane, Barry Smyth, and Padraig Cunningham. Systems, tasks, and adaptation knowledge. In Manuela Veloso and Agnar Aamodt, editors, *International Conference on Case-Based Reasoning*, Lecture Notes in Artificial Intelligence, pages 461–470, Sesimbra, Portugal, 1995. Springer, Verlag.

[21] Thomas R. Hinrichs. Strategies for adaptation and recovery in design problem solver. In *Workshop on case-based Reasoning, DARPA 89*, pages 115–118. Morgan-Kaufmann, San Mateo, 1989.

[22] K. Hua, B. Faltings, and I. Smith. Cadre: Case-based geometric design. *Journal of Artificial Intelligence in Engineering*, 10:171–183, 1996.

[23] Jana Koehler. Planning from second principles. *Artificial Intelligence*, 87:145–186, 1996.

[24] Janet Kolodner. Judging which is the "best" case for a case-based reasoner. In *DARPA Case-Based Reasoning Workshop*, pages 77–81. Morgan Kaufmann, San Mateo, CA, 1989.

[25] Janet Kolodner. *Case Based Reasoning.* Morgan Kaufman Publishers, 1993.

[26] Janet Kolodner, R. Simpson, and Katia Sycara-Cyranski. A process model of case-based reasoning in problem solving. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, volume 1, pages 284–290, Los Angeles, CA, 18th-23rd August 1985 1985.

[27] Janet L. Kolodner. Reconstructive memory: A computer model. *Cognitive Science*, 7:281–328, 1983.

[28] David B. Leake, editor. *Case-Based Reasoning: Experiences, Lessons and Future Directions.* AAAI Press, 1996.

[29] Jean Lieber. Reformulations and Adaptation Decomposition. In J. Lieber, E. Melis, A. Mille, and A. Napoli, editors, *Formalisation of Adaptation in Case-Based Reasoning.* Third International Conference on Case-Based Reasoning Workshop, ICCBR-99 Workshop number 3, S. Schmitt and I. Vollrath (volume editor), LSA, University of Kaiserslautern, 1999.

[30] Mary Lou Maher, M. Bala Balachandran, and Dong Mei Zhang. *Case-based reasoning in design.* Lawrence Erlbaum Associates, 1995.

[31] Erica Melis, Jean Lieber, and Amedeo Napoli. Reformulation in Case-Based Reasoning. In B. Smyth and P. Cunningham, editors, *Fourth European Workshop on Case-Based Reasoning, EWCBR-98*, Lecture Notes in Artificial Intelligence 1488, pages 172–183. Springer, 1998.

[32] E. Plaza and J.-L. Arcos. Constructive adaptation. In S. Craw and A. Preece, editors, *Proc. 6th ECCBR 2002*, volume Advances in Case-Based Reasoning of *Lecture Notes on Artificial Intelligence 2416*, pages 306–320. Springer-Verlag, Berlin, 2002.

[33] Pearl Pu and Lisa Purvis. Formalizing the adaptation process for case-based design. In Mary Lou Maher and Pearl Pu, editors, *Issues and Applications of Case-Based Reasoning in Design*, pages 221–240. Lawrence Erlbaum Associates, 1997.

[34] Michael Richter. Classification and learning of similarity measures. In Clas Opiz and Lausen Klar, editors, *Proceedings der Jahrestangung der Gesellschaft für Klassifikation*, Studies in Classification, Data Analysis and Knowledge Organisation. Springer Verlag, 1992.

[35] Roger C. Schank. *Dynamic Memory: A theory of reminding and learning in computers and people.* Cambridge University Press, 1982.

[36] Roger C. Schank and Christopher K. Riesbeck. *Inside Case Based Reasoning.* LEA Publishers, Hillsdale, New Jersey07642, 1989.

[37] Manuela M. Veloso, Héctor Munoz-Avila, and Ralph Bergmann. General-purpose case-based planning: Methods and systems. *AI Communications*, 9(3):128–137, 1996.

7 Annex: The Prolabo application

7.0.4 An illustration of the digestion process

The following figures 9, 10, 11, 12 and 13, proposes views of the specification of a digestion problem, and a digestion solution, through the application interface.

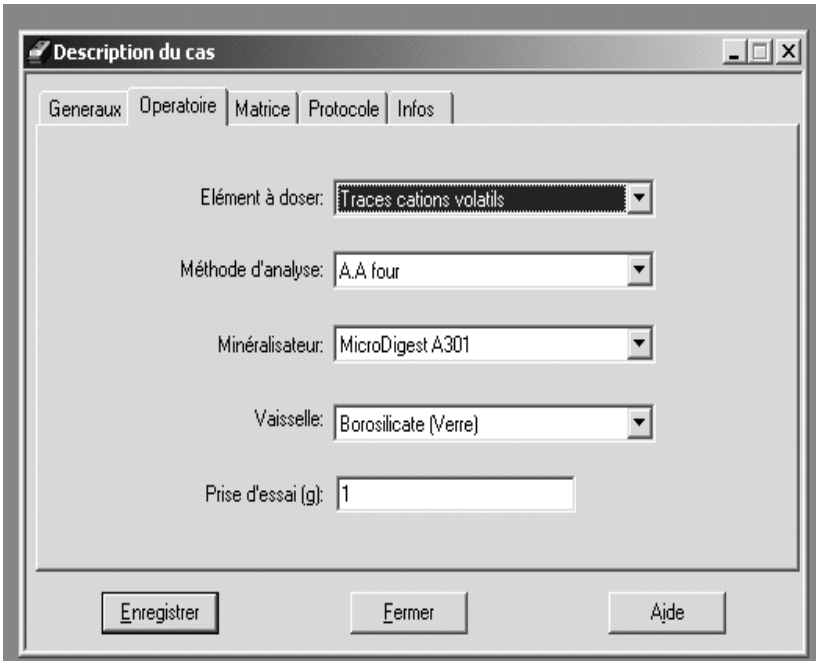

Figure 9: An example of a description of a digestion specification (part 1 of the problem).

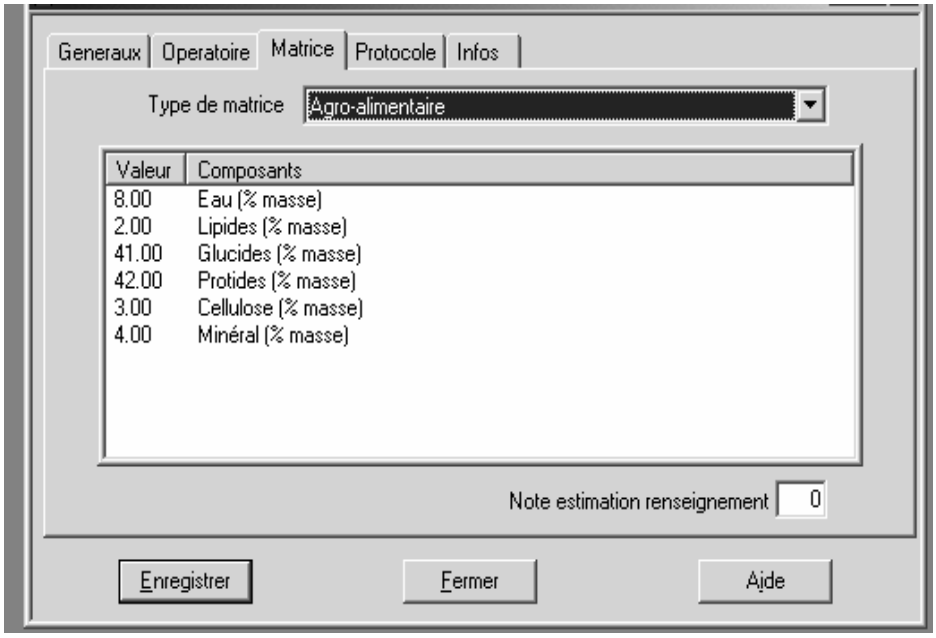

Figure 10: An example of a description of a digestion specification (part 2 of the problem).

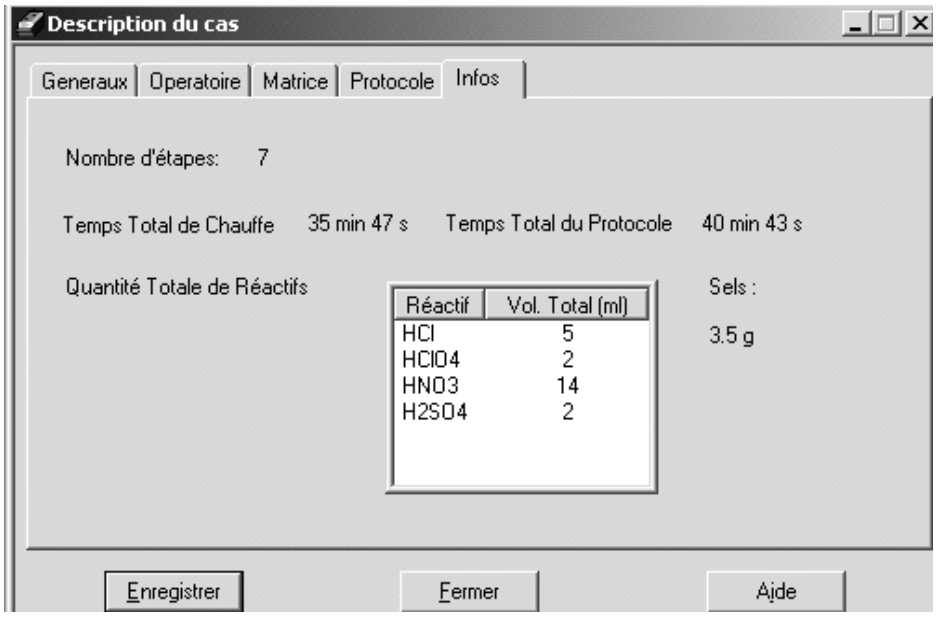

Figure 11: An example of a description of a digestion solution (numerical balance).

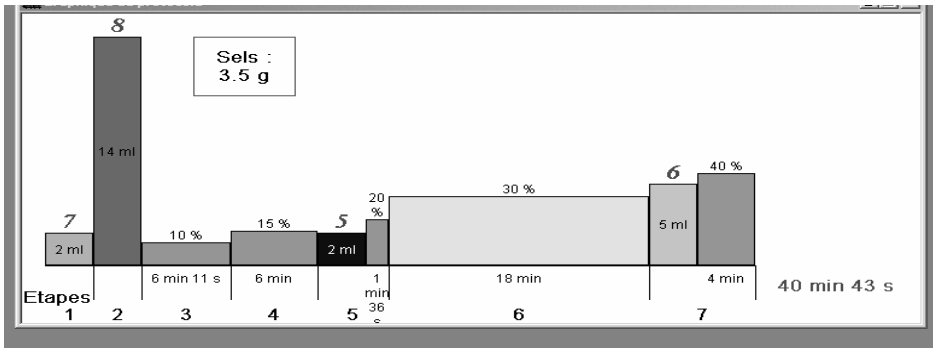

Figure 12: An example of a description of a digestion solution (graphical balance).

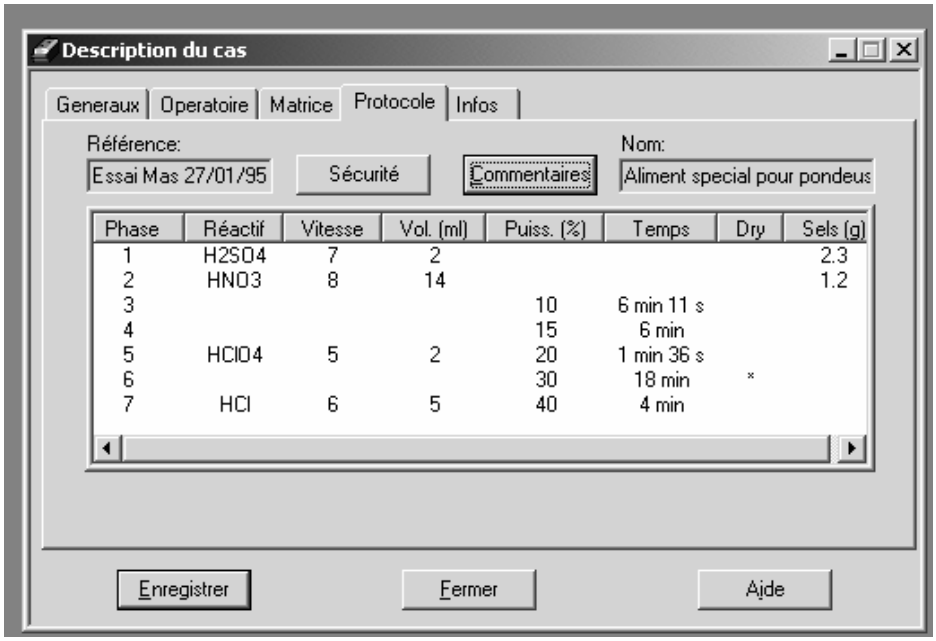

Figure 13: An example of a description of a digestion solution (the whole program).