Differential Adaptation: an Operational Approach to Adaptation for Solving Numerical Problems with CBR

Béatrice Fuchs[a], Jean Lieber[b], Alain Mille[c], Amedeo Napoli[d]

[a]*LIRIS (CNRS - Université de Lyon), Université Lyon 3, 6 rue Rollet, 69008 Lyon, France*
beatrice.fuchs@liris.cnrs.fr
[b]*LORIA (CNRS – INRIA NGE – Université de Lorraine), Orpailleur Research Group, B.P. 239, 54506 Vandœuvre-lès-Nancy, France*
Jean.Lieber@loria.fr
[c]*LIRIS (CNRS – Université de Lyon), Nautibus, 8 bd Niels Bohr, 69100 Villeurbanne, France*
amille@liris.cnrs.fr
[d]*LORIA (CNRS – INRIA NGE – Université de Lorraine), Orpailleur Research Group, B.P. 239, 54506 Vandœuvre-lès-Nancy, France*
Amedeo.Napoli@loria.fr

Abstract

Case-based reasoning relies on four main steps: retrieval, adaptation, revision and retention. This article focuses on the adaptation step; we propose *differential adaptation* as an operational formalization of adaptation for numerical problems. The solution to a target problem is designed on the basis of relations existing between a source case (problem and solution) and a target case. Differential adaptation relies on the metaphor of differential calculus where small variations on variable values are related to variations of function values. Accordingly, variations between problems correspond to variations between variable values and variations between solutions to variations between function values. Operators inspired from differential calculus are able to manipulate the variations and to support the whole adaptation process. Differential adaptation is operational and provides generic operators that can be reused for different real-world numerical situations.

Keywords: adaptation, case-based problem solving, knowledge intensive CBR system, reformulation, similarity path, differential calculus

1. Introduction

Case-based reasoning (CBR) is widely used in many applications and tasks to solve problems in weak theory domains. CBR solves new problems by retrieving and adapting solutions of previously solved cases stored in a case base [35, 23, 25]. The basic knowledge unit is the case which represents a problem solving episode. A typical CBR cycle is composed of four main steps. Given a new problem, called a *target problem*, the *retrieval* step selects a *source case* from a case base. The *adaptation* step builds a solution to the target problem, relying on the differences between the source and target problems, and modifying accordingly the solution of the retrieved source case. The *revision* step improves the current target solution depending on actual and expected system results. Finally the *retention* step stores the newly solved case for further reuse.

In *knowledge intensive* CBR systems, cases are completed with domain knowledge which provides case vocabulary and semantics for interpreting situations, assessing similarity and supporting the adaptation process. Knowledge intensive CBR systems rely on several knowledge containers [34]: cases, domain knowledge, similarity measures, and adaptation knowledge. More generally, it is usually assumed in CBR that it is possible to rely on adaptation knowledge and variations between problems for finding and controlling variations between solutions: the less there are variations between problems the less there are variations between solutions. Nevertheless, modeling adaptation is a difficult task as it takes into account domain knowledge not always available or easy to acquire [15]. Most of the time, in practical applications, the adaptation step is either restricted to a simple copy of the source solution without any modification, or implemented in an *ad hoc* way in the framework of the application.

Adaptation in CBR has been studied in various situations, e.g. [21, 38, 32, 37, 15], and there does not yet exist a general model of adaptation. By contrast, this article proposes *differential adaptation* as an operational approach to adaptation for solving numerical problems with CBR, i.e. problems where variables have numerical or totally ordered values. Adaptation is based on relations existing, on the one hand, between a source problem and a target problem, and on the other hand, between a source problem and the source solution. Differential adaptation materializes the research results of studies on adaptation within the CBR process, carried out by the authors for several years, e.g. [24, 18, 19, 12], and more recently [13, 10, 17].

1.1. Adaptation and dependencies

Let us assume that we are given a target problem `tgt` and a case base including a collection of source cases `srce-case` = (`srce`, `Sol`(`srce`)), where `srce` denotes a source problem and `Sol`(`srce`) the associated solution. During the retrieval step, several source problems of the case base are retrieved and compared to the target problem. The matching process locally establishes the correspondence between a feature or a descriptor of the source problem and a corresponding feature of the target problem. The similarities and differences between the local features of the source and target problems are made explicit. Then, a global similarity measure combines the local feature similarities for assessing a global similarity between the source and the target problem. Accordingly, a general schema for adaptation is proposed in Figure 1, where we have the following correspondences:

(i) 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. *Similarity assessment (or matching)* matches the descriptor of `srce` to descriptors of `tgt` and records the differences between problem descriptors.

(ii) 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, A *dependency* records the links between problem descriptors and solution descriptor. *Solution modification* combines the result of the two previous operations to build a solution `Sol`(`srce`) from a modification of the source problem.

Indeed, CBR tries to find and to control variations between solution features from variations between problem features. The task of adaptation is to rely on the selected source case for designing a solution to the target problem. Hereafter, we present the principles of differential adaptation that applies to numerical problems, i.e. problems with numerical descriptors. Differential adaptation relies on a simple and intuitive basis and is useful when dependencies between descriptors are known and can be controlled.

1.2. Contribution of the paper

In terms of variations, the differences between problem descriptors and the dependencies between problem and solution descriptors can be modeled

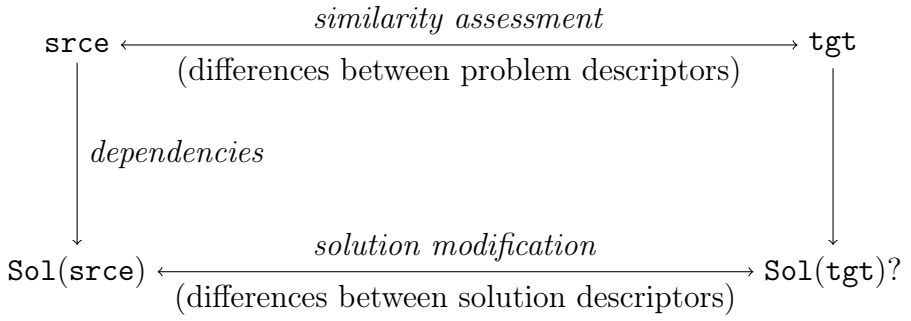


Figure 1: A three-phase schema of the adaptation process.

and computed through formulas analogous to differential calculus formulas [14]. Accordingly, in this paper, we introduce *differential adaptation* as a framework for adaptation inspired from differential calculus and based on the following main ideas:

- The adaptation process is decomposed into a sequence of adaptation steps materialized by *reformulations* [27]. Actually, this decomposition is based on a sequence of intermediate problems between the source and the target problems –in the problem space– called a *similarity path* (see Figure 2).

- A reformulation relies on a local operator that derives a problem feature from another problem feature (relations $\mathtt{r}^{\mathtt{k}}$ in Figure 2), and on a local operator transforming a solution feature into another solution feature (relations $\mathcal{A}_{\mathtt{r}^{\mathtt{k}}}$ in Figure 2).

- The similarity path has a corresponding path in the solution space, called an *adaptation path*, that modifies the source solution to build a target solution (see Figure 2). The local operators in the similarity path have corresponding local operators in the adaptation path, for adapting the source solution into a target solution.

A set of differences characterizes the global difference between the source and target problems, while a set of dependencies materializes the relations between operators in the space of problems and the corresponding operators in the space of solutions. The metaphor chosen for modeling the dependencies

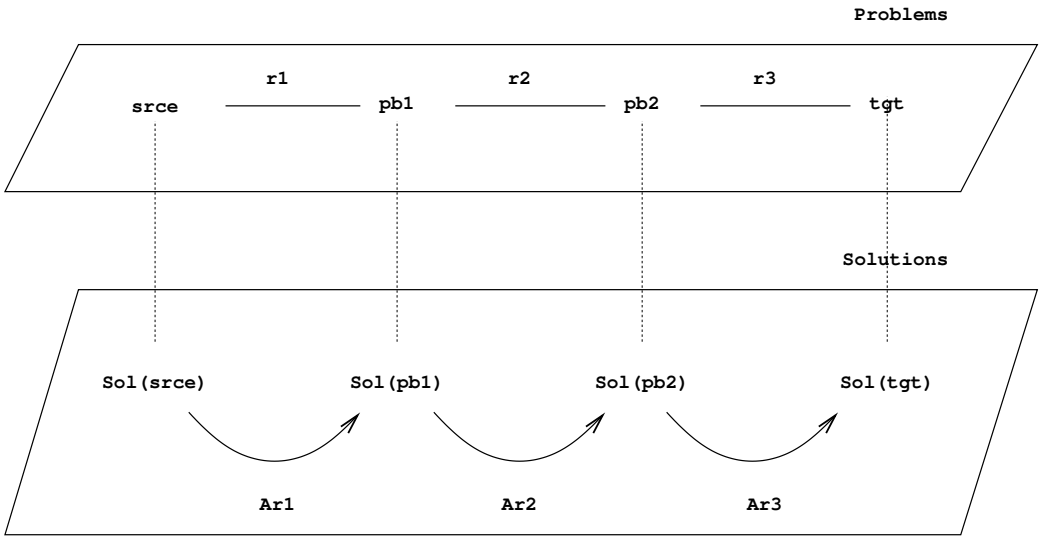


Figure 2: A sequence of reformulations relating a source problem and its solution to a target problem and its solution.

between the problem and its solution, and between the source and target problems, relies on differential calculus and partial derivatives that represent dependencies.

The differential adaptation model is suited to numerical problems and relies on differential calculus, similarity paths and adaptation paths. An objective of this paper is to show that differential adaptation provides an operational and generic model for solving numerical problems with CBR. Differential adaptation can be applied to any numerical problem provided that the dependencies between descriptors can be computed and controlled, in the same way as for computing the derivative of a function we should know the rules for computing the derivatives of elementary functions.

The outline of the article is as follows. First, the framework of differential adaptation illustrated by concrete examples is given. Then, Prolabo experiment shows how differential adaptation may be implemented in a real-world application. A discussion on the applicability and limitations of differential adaptation follows. Then, experiments involving the approximation of mathematical functions show the good behavior of differential adaptation. Finally, related work is discussed and the paper is concluded.

2. The principles of differential adaptation

2.1. A concrete and motivating example

For illustration, let us consider the following example[1]: 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 (the source case consists of a source problem and a 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 product. The differences between the features of the source and the target problems are 0 for the first feature 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 0 for the first operand (namely 4 and the variation is $4 - 4 = 0$) and $4 \times 1 = 4$ for the second operand (namely 5 and the variation is $5 - 4 = 1$). The adapted solution feature will be given by the source solution feature and the effect of the variation using the $+$ operator, i.e. $4 \times 5 = 16 + 0 + 4 = 20$.

2.2. The two principles of differential adaptation

In differential calculus [14], the "total derivative" of a function of several variables can be expressed through the a sum of partial derivatives. Considering $y \in \mathbb{R}^n$ as a function of $x \in \mathbb{R}^m$, the following formulas give the total derivative of y, denoted by dy, w.r.t. dx and the partial derivatives:

$$dy_j = \sum_{i=1}^{m} \frac{\partial y_j}{\partial x_i}(x) \times dx_i \tag{1}$$

where $f : x \mapsto y$ is a differentiable function from $\mathbb{R}^m$ to $\mathbb{R}^n$

$$dx = (dx_1, dx_2, \ldots, dx_m)$$

$$dy = (dy_1, dy_2, \ldots, dy_n)$$

[1] Inspired from a talk given by Ian Watson at ICCBR-99.

The total derivative formula adds all *indirect dependencies* of y_j on x_i for computing the overall dependency of y on x. Then, if dx_i and dy_j are interpreted as "small variations" of x_i and y_j, then (1) can be considered as an approximation: given $x^0, x^1 \in \mathbb{R}^m$ such that x^0 is similar to x^1, and $y^0 = f(x^0)$, equation (1) allows to derive $y^1 \simeq f(x^1)$ knowing $\frac{\partial y_j}{\partial x_i}(x^0)$, and taking $dx_i = x_i^1 - x_i^0$ and $dy_j = y_j^1 - y_j^0$. Here, $y \simeq f(x)$ has to be interpreted as "y is a solution or an approximation of the solution of problem x" where f is a function associating solution y to problem x.

Equation (1) may also be interpreted in the context of adaptation in CBR. Let us consider a domain where the problem space is $\mathbb{R}^m$, the solution space is $\mathbb{R}^n$, and where $y \in \mathbb{R}^n$ solves $x \in \mathbb{R}^m$ with $y \simeq f(x)$. Then, equation (1) can be used to adapt the solution y^0 of the problem x^0 into a solution y^1 of the problem x^1. In other words, $\frac{\partial y_j}{\partial x_i}(x^0)$ defines the *influence* of variations of x_i on variations of y_j in the context of x^0. Equation (1) provides the *first principle of differential adaptation.*

Now, given $y^1 \simeq f(x^1)$ and x^2 similar to x^1, (1) can be used to compute $y^2 \simeq f(x^2)$. More generally, if $x^{k-1} \simeq x^k$ for $1 \leq k \leq q$, then $y^q \simeq f(x^q)$ can be computed by applying the Equation (1) q times:

1. $(x^0, y^0, x^1) \mapsto y^1$,
2. $(x^1, y^1, x^2) \mapsto y^2$,

   ...

q. $(x^{q-1}, y^{q-1}, x^q) \mapsto y^q$,

which can be likened to Euler's method for solving numerical differential equations. Thus, the adaptation of the solution y^0 of a source problem x^0 to solve a target problem x^q can be decomposed into several steps by firstly finding intermediate problems x^1, x^2, ..., x^{q-1} such that $x^0 \simeq x^1$, $x^1 \simeq x^2$, ..., $x^{q-1} \simeq x^q$, and secondly, applying equation (1) q times. This constitutes the *second principle of differential adaptation* leading to *multi-step differential adaptation.*

Back to multiplication of integers. Let us consider the following simple numerical problem: "$3 \times 5 \simeq$?" knowing that $4 \times 4 = 16$, where $\times$ stands for multiplication of integers. The application of the notations introduced

above gives $f : x = (x_1, x_2) \in \mathbb{R}^2 \mapsto y = x_1 \times x_2 \in \mathbb{R}$ ($m = 2$, $n = 1$), $x^0 = (4, 4)$, $y^0 = 16$, and $x^q = (3, 5)$. The partial derivatives are $\frac{\partial y}{\partial x_1}(x) = x_2$ and $\frac{\partial y}{\partial x_2}(x) = x_1$.

The differential adaptation of (x^0, y^0) to solve x^q gives, with $dx_1 = x_1^1 - x_1^0$ and $dx_2 = x_2^1 - x_2^0$ and without introducing any intermediate problem, i.e. setting $q = 1$:

$$\begin{aligned} y^1 &= y^0 + \frac{\partial y}{\partial x_1}(x^0) \times (x_1^1 - x_1^0) + \frac{\partial y}{\partial x_2}(x^0) \times (x_2^1 - x_2^0) \\ &= 16 + x_2^0 \times (3 - 4) + x_1^0 \times (5 - 4) = 16 \end{aligned}$$

As a first result, we obtain $3 \times 5 \simeq 16$ where 16 is an approximation of the actual result of the operation. Then using multi-step differential adaptation, i.e. setting $q = 2$ and introducing the intermediate problem $x^1 = (3, 4)$, the differential adaptation of (x^0, y^0) to solve x^q gives, with $dx_1 = x_1^1 - x_1^0$ and $dx_2 = x_2^1 - x_2^0$:

$$\begin{aligned} y^1 &= y^0 + \frac{\partial y}{\partial x_1}(x^0) \times (x_1^1 - x_1^0) + \frac{\partial y}{\partial x_2}(x^0) \times (x_2^1 - x_2^0) \\ &= 16 + x_2^0 \times (3 - 4) + x_1^0 \times (4 - 4) = 16 - 4 = 12 \\ y^2 &= y^1 + \frac{\partial y}{\partial x_1}(x^1) \times (x_1^2 - x_1^1) + \frac{\partial y}{\partial x_2}(x^1) \times (x_2^2 - x_2^1) \\ &= 12 + x_2^1 \times (3 - 3) + x_1^1 \times (5 - 4) = 15 \end{aligned}$$

In this second case, we obtain $3 \times 5 \simeq 15$. This shows that the introduction of intermediate problems may have an influence on the quality of the solution in the differential adaptation process.

2.3. Case representation and notations

In the following, we assume that `Problems` and `Solutions` respectively denote the problem and solution spaces. A problem x (resp. a solution y) is by definition an element of `Problems` (resp. `Solutions`). There exists a binary relation on `Problems` $\times$ `Solutions` interpreted as "has for solution". This relation is not completely known, except for some of its instances (x^0, y^0) which correspond to the *source cases* in the case base.

We assume that a problem x is represented by a set of m descriptors, i.e. $x = (x_1, x_2, \ldots, x_m)$, where x_i is the i^{th} descriptor of x and x_i belongs to a $\mathbb{R}$. Similarly, for $y \in \texttt{Solutions}$, $y = (y_1, y_2, \ldots, y_n)$. Therefore $\texttt{Problems} = \mathbb{R}^m$ and $\texttt{Solutions} = \mathbb{R}^n$. All problems (resp. all solutions) are supposed to be of the same "size" m (resp. n). If a_i denote the i^{th} projection, i.e. $a_i : x = (x_1, x_2, \ldots, x_m) \in \texttt{Problems} \mapsto x_i$, then a_1, a_2, ..., a_m are the problem attributes. In a similar way, the j^{th} projection of `Solutions` is a solution attribute with range $\mathbb{R}$. This representation follows an attribute-value formalism [23, 7].

2.4. From source to target

This section makes precise the "horizontal view" of Figure 1, i.e., the relationships between source and target cases. These relationships are studied firstly as a one-step approach and then as a composition of steps.

2.4.1. One-step similarity assessment and solution modification

Similarity assessment: computing dx. Given x^0 and x^1, a source problem and a target problem, dx represents the relationships between x^0 and x^1 that are used in the adaptation process: dx represents some of the similarities and dissimilarities between x^0 and x^1. The process $(x^0, x^1) \mapsto dx$ is called the *similarity assessment*. We assume that this process can be split into a set of local processes, one for each problem descriptor:

$$dx = (dx_1, dx_2, \ldots, dx_m)$$

and $(x_i^0, x_i^1) \mapsto dx_i$ is the similarity assessment of the i^{th} problem descriptor.

The computation of dx_i depends on the difference $dx_i = x_i^1 - x_i^0$. Thus, we have the following mapping: $(x_i^0, x_i^1) \in \mathbb{R} \times \mathbb{R} \mapsto dx_i = x_i^1 - x_i^0 \in \mathbb{R}$ with $x_i^1 - x_i^0 = 0$ iff $x_i^0 = x_i^1$.

Solution modification: computing dy. $dy = (dy_1, dy_2, \ldots, dy_n)$ is defined similarly to dx, with dy_i is the operation taking as parameters $y_j^0, y_j^1 \in \mathbb{R}$ and giving as output $dy_j = y_j^1 - y_j^0 \in \mathbb{R}$. The difference between dx and dy is that, in the adaptation process, x^0 and x^1 are available before dx, but dy is available before y^1. In other words, the adaptation process includes the phases $(x^0, x^1) \mapsto dx$ and $(y^0, dy) \mapsto y^1$, the computation of dy being detailed in the next section.

Now, given y^0 and dy, how should y^1 be computed? The answer is based on the fact that y_j^1 is a solution of the equation $dy_j = y_j^1 - y_j^0$. This can be

alternatively stated as: $(y_j^0, dy_j) \in \mathbb{R} \times \mathbb{R} \mapsto y_j^1 \in \mathbb{R}$ such that $y_j^1 = y_j^0 + dy_j$ iff $y_j^1 - y_j^0 = dy_j$. When the above equation has no solution, the adaptation process fails: (x^0, y^0) cannot be adapted to solve x^1.

2.4.2. Similarity assessment and solution modification within several steps

According to the second principle of differential adaptation, adaptation can be decomposed in q steps: given $q \geq 1$, (x^0, y^0) a source case, and x^q a target problem, the similarity assessment aims at finding, for each $k \in \{1, 2, \ldots, q-1\}$, $x^k = (x_1^k, x_2^k, \ldots, x_m^k)$ and $dx^k = (dx_1^k, dx_2^k, \ldots, dx_m^k)$ such that $dx_i^k = x_i^k - x_i^{k-1}$ for $i \in \{1, 2, \ldots, m\}$.

From each dx^k, $dy^k = (dy_1^k, dy_2^k, \ldots, dy_n^k)$ can be computed (as detailed in the next section) and the last phase of differential adaptation aims at solving the system of equations:

$$dy_j^k = y_j^k - y_j^{k-1} \qquad 1 \leq j \leq m \text{ and } 1 \leq k \leq q$$

given y_j^0 and dy_j^k for any j and any k. In particular:

$$dy_j^q = (\ldots((y_j^0 + dy_j^1) + dy_j^2) + \ldots + dy_j^{q-1})$$

And this finally provides a n-tuple value $y^q = (y_1^q, y_2^q, \ldots, y_n^q) \in \texttt{Solutions}$.

2.5. From problem to solution

This section makes precise the "vertical view" of Figure 1, i.e. the relationships between problems and solutions. More precisely, the question is to study how the solution descriptors change when the problem descriptors change, i.e. how to compute $dy = (dy_1, dy_2, \ldots, dy_n)$ knowing $dx = (dx_1, dx_2, \ldots, dx_m)$? The question can be rephrased as follows:

- For each $i \in \{1, 2, \ldots m\}$ and $j \in \{1, 2, \ldots n\}$, what is the contribution of the variation dx_i of the problem descriptor x_i to the variation dy_j of the solution descriptor y_j, when every other problem descriptor remain constant? This contribution is denoted by $d_i y_j$ and depends on the source problem x^0.

- Given the contributions $d_i y_j$ of each problem descriptor x_i to the variation of the solution descriptor y_j, the variation dy_j is computed as follows, where $\mathbb{R}$ is the domain of values of y_j and also the set of possible differences for y_j:

$$(d_1 y_j, d_2 y_j, \ldots, d_m y_j) \in \mathbb{R}^m \mapsto dy_j \in \mathbb{R} \qquad (2)$$

Computing the contribution d_iy_j to dy_j. Considering a function $f : x \in \mathbb{R}^m \mapsto y = f(x) \in \mathbb{R}^n$ and equation (1), the contribution of d_iy_j is given by:

$$d_iy_j = \frac{\partial y_j}{\partial x_i}(x^0) \times dx_i \tag{3}$$

More generally, we consider that $\left(\frac{\partial y_j}{\partial x_i}(x^0)\right)$ is the function mapping dx_i to d_iy_j. When $x_i^0 = x_i^1$, then $dx_i = 0$ and it comes:

$$\frac{\partial y_j}{\partial x_i}(x^0) \times 0 = \left(\frac{\partial y_j}{\partial x_i}(x^0)\right)(0) = 0 \tag{4}$$

The knowledge of $x^0 \mapsto \frac{\partial y_j}{\partial x_i}(x^0)$, i.e. the influence of variations of x_i on the variations of y_j in the context of x^0, constitutes the main part of the adaptation knowledge and can take different forms.

Computing dy_j from the contributions d_iy_j. Based on the preceding developments, there exists a function with the profile (2), merging the contributions d_iy_j as follows:

$$dy_j = d_1y_j + d_2y_j + d_3y_j + \ldots + d_my_j = \sum_{i=1}^{m} d_iy_j$$

2.6. An algorithm for differential adaptation

We present in Figure 3 an algorithm which summaries the operations involved in differential adaptation (multi-step process). In the next section, we show how differential adaptation and this algorithm are used to solve real-world numerical problems, based on the principles introduced and explained in this section.

3. The Prolabo application

3.1. An overview of the Prolabo application

Prolabo is a company manufacturing and marketing devices for chemical, pharmaceutical, biochemical, and biological laboratories. One of these devices is a guided microwave digestion device –"digester" for short– in charge of a programmed digestion process. The role of such a digestion process is

/* Multi-step differential adaptation */
/* Solving problem x^q by reusing source case (x^0, y^0) */

Inputs :

- (x^0, y^0) a source case with $x^0 = (x^0_1, \ldots, x^0_m)$ and with $y^0 = (y^0_1, \ldots, y^0_n)$
- x^q a target problem
- $\{x^k\}_{k=1\ldots q-1}$ a sequence of intermediate problems with $x^k = (x^k_1, \ldots, x^k_m)$
- $\left\{\frac{\partial y_j}{\partial x_i}(x^k)\right\}_{i=1\ldots m, j=1\ldots n, k=1\ldots q-1}$ a set of dependencies

Output: y^q a solution of the target problem x^q

begin
  for $k \leftarrow 1$ ***to*** q **do**
    /* For each step of the similarity path */
    for $j \leftarrow 1$ ***to*** n **do**
      /* For each solution descriptor y^k_j, compute local influences $d_i y^k_j$ for each dependency */
      for $i \leftarrow 1$ ***to*** m **do**
        $dx^k_i \leftarrow x^k_i - x^{k-1}_i$
        $d_i y^k_j \leftarrow \frac{\partial y_j}{\partial x_i}(x^{k-1}) \times dx^k_i$
      end
      /* Combine local influences $d_i y^k_j$ into a global variation dy_j */
      $dy^k_j \leftarrow \sum_{i=1\ldots m} d_i y^k_j$
      $y^k_j \leftarrow y^{k-1}_j + dy^k_j$
    end
  end
  return y^q /* with $y^q = (y^q_1, y^q_2, \ldots, y^q_n)$ */
end

Figure 3: A multi-step algorithm for differential adaptation.

to prepare –from a product to be analyzed– a sample for chemical analysis, thanks to various analysis processes (see Figure 4). After digestion, the sample must be constituted only of the chemical atomic elements of the product to be analyzed. The digester is in charge of breaking all molecular bonds between atoms of the sample either chemically with aggressive chemical agents, or physically with mechanical or thermal microwave effects. The digestion process is fully automated and depends, on the one hand, on injection pumps controlling the special chemical agent injection, and, on the other hand, on a magnetron controlling the microwave effects. Injection pumps are controlled w.r.t. three main parameters: i.e. the chemical agents to be injected, the injection speed, and the injection duration. Meanwhile, there are two main parameters for controlling the magnetron: i.e. the power value and the

emission duration.

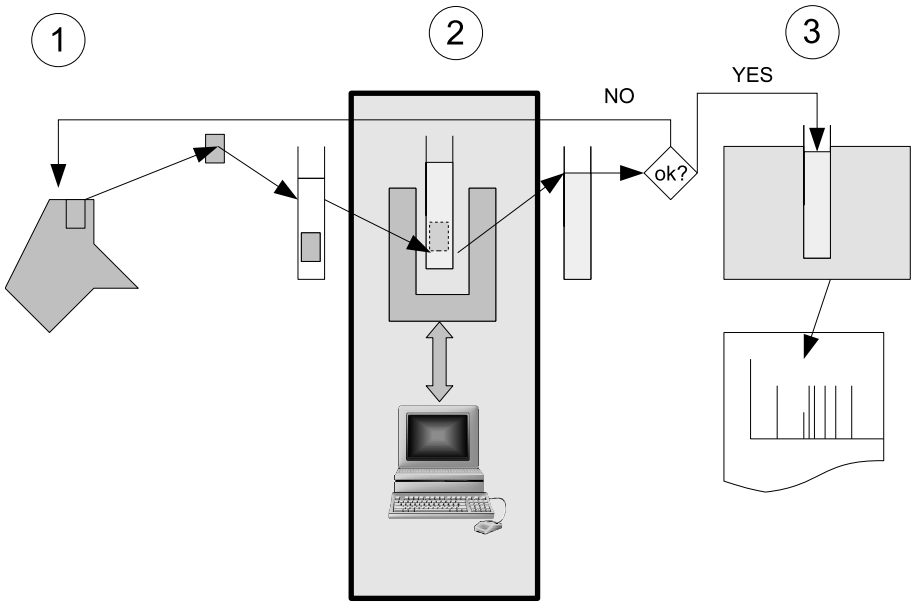


Figure 4: The digestion process for chemical analysis: (1) A product sample is selected and inserted in the guided microwave digester. (2) The digester decomposes the molecules of the sample into atomic elements. If the digestion process succeeds: (3) The analysis can be processed and provides atomic element spectrums. Otherwise, the process returns to step (1).

A digestion program is composed of sequential steps –from 5 up to 20 steps– where each step is controlled by a set of parameters, e.g. choice of the chemical agents to be injected, injection speed of the chemical agents, injection duration, magnetron power (percent of the magnetron maximum power), and magnetron powering duration (only one step is considered hereafter).

Programming a microwave digester remains a difficult and costly task, because it is manual and iterative. Thus, laboratory engineers try to minimize the number of trials by reusing past experience of successful digestion cases for similar samples in similar analysis conditions. This explains why CBR can be useful for solving the design of a digestion program. Hereafter, we show how differential adaptation can be used to solve such a problem. Accordingly, the so-called "Prolabo application" is based on the adaptation of previous "digestion programs" in order to design new digestion programs. The problem of designing a digestion program is considered as a planning

problem with numerical descriptors and is solved with a CBR approach based on differential adaptation.

3.2. Knowledge elements within a case

A "digestion case" includes the description of a problem and its solution. The problem part describes the context of a digestion and the descriptors to be used for adapting a digestion program. The solution part is composed of a synthesis of the digestion program, called a "digestion plan", that can be effectively processed by a digester.

Table 1 shows problem descriptors where, actually, only two out of thirteen descriptors are considered in this example: ***Analyzer Type*** (x_1) is related to the analyzer in charge of digestion; ***Analysis Class*** (x_7) gives information about the volatility of atomic elements to be tracked. Table 2 shows solution descriptors and only one of them is considered in the following: ***Moderation Level*** (y_2) is an integer measuring the moderation of the digestion process by gradually heating.

The current problem can be stated as follows: given ***Analyzer Type*** (x_1) and ***Analysis Class*** (x_7), can we compute ***Moderation Level*** (y_2) w.r.t. a source case which has been retrieved and provides values for x_1 and x_7? On a practical level, the higher the analysis class –elements to be analyzed are more volatile– the more the digestion plan has to be moderated. In addition, the more the analyzer is sensitive to aggressive chemical agents the more the digestion plan should be moderated with respect to the provided energy.

Then, for a given target problem and a retrieved source case, adaptation is performed on the source solution according to the differences between the source and target problems. Firstly, let us consider the dependency between ***Analysis Class*** (x_7) and ***Moderation Level*** (y_2) for the digestion plan. The influence of the variations of ***Analysis Class*** (from x_7^0 to x_7^1) on the variations of ***Moderation Level*** (y_2) in the context of the source problem (x^0) is given by $\frac{\partial y_2}{\partial x_7}(x^0)$. The variation of y_2 due to the variation of x_7 is given by $d_7 y_2 = \frac{\partial y_2}{\partial x_7}(x^0) \times dx_7$ (here $\times$ stands for multiplication). The different sets of values relating the descriptors ***Analysis Class*** (x_7) and ***Moderation Level*** (y_2) are based on domain knowledge and given as follows:

- $V_7 = \{1, 2, 3, 4, 5\}$ is the domain of values for the descriptor ***Analysis Class*** (x_7).

Digestion problem statement		
Id	Name	Type
x_1	*Analyzer Type*	Integer
x_2	Test Tube Nature	Integer
x_3	Injection Speed	Real
x_4	Magnetron Power	Real
x_5	Tube Capacity	Real
x_6	Max Power Gradient	Real
x_7	*Analysis Class*	Integer
x_8	Sample Weight	Real
x_9	Lipids Quantity	Real
x_{10}	Glucide Quantity	Real
x_{11}	Mineral Quantity	Real
x_{12}	Cellulose Quantity	Real
x_{13}	Water Quantity	Real

Table 1: The descriptors of a problem and their characteristics. *Analyzer Type* (x_1) and *Analysis Class* (x_7) are the descriptors used in the running example.

Digestion program description		
Id	Name	Type
y_1	Plan Type	Integer
y_2	*Moderation Level*	Integer
y_3	Step Number	Integer
y_4	Total Duration	Real
y_5	Total Energy	Real
y_6	Injected Product	Integer
y_7	Injected Quantity	Real

Table 2: The descriptors of a solution (only one step is considered). *Moderation Level* (y_2) is the descriptor adapted in the running example.

- $\Delta V_7 = \{-2, -1, 0, +1, +2\}$ is the domain of value variations for x_7, i.e. the set of possible differences between two values of x_7.
- $W_2 = \{1, 2, 3, 4, 5, 6\}$ is the domain of values for *Moderation Level* (y_2).
- $\Delta W_2 = \{-3, -2, 0, +2, +3\}$ is the domain of value variations for y_2.

Each variation in ΔV_7 is mapped to a corresponding variation in ΔW_2

giving the value of $\frac{\partial y_2}{\partial x_7}$. For example, when the variation for x_7 is of -2 in ΔV_7, there is a variation for y_2 of -3 in ΔW_2 and $\frac{\partial y_2}{\partial x_7} = 3/2$.

Now, let us consider that, in the current problem, $x_7^0 = 3$ (*Analysis Class* in source problem), $x_7^1 = 4$ (*Analysis Class* in target problem), and $y_2^0 = 3$ (*Moderation Level* in source solution). Then the value of y_2^1 (*Moderation Level* in target solution) is computed as follows:

$y_2^1 = y_2^0 + d_7 y_2$ where $d_7 y_2 = \frac{\partial y_2}{\partial x_7}(x^0) \times dx_7$,

$dx_7 = x_7^1 - x_7^0 = 4 - 3 = 1$ and $+1$ in ΔV_7 corresponds to $+2$ in ΔW_2,

thus $\frac{\partial y_2}{\partial x_7}(x^0) = 2/1 = 2$ and then,

$y_2^1 = y_2^0 + \frac{\partial y_2}{\partial x_7}(x^0) \times dx_7 = 3 + 2 = 5$,

meaning that the adapted value is 5 for *Moderation Level* in the solution of the target problem.

Secondly, let us consider the dependency between x_1 (*Analyzer Type*) and y_2 (*Moderation Level*) for the digestion plan. The influence of the descriptors x_1^0 and x_1^1 (*Analyzer Type*) on the descriptor y_2 (*Moderation Level*) in the context of the source problem x^0 is denoted by $\frac{\partial y_2}{\partial x_1}(x^0)$. The adaptation of y_2 (*Moderation level*) is given by $d_1 y_2 = \frac{\partial y_2}{\partial x_1}(x^0) \times dx_1$.

Let us consider the different domains of values relating the descriptors x_1 (*Analyzer Type*) and y_2 (*Moderation Level*), being still based on domain knowledge:

- $V_1 = \{1, 2, 3, 4, 5\}$ is the domain of values for x_1 (*Analyzer Type*).
- $\Delta V_1 = \{-2, -1, 0, +1, +2\}$ is the domain of value variations for x_1.
- $W_2 = \{1, 2, 3, 4, 5, 6\}$ is the domain of values for y_2 (*Moderation Level*).
- $\Delta W_2 = \{-2, -1, 0, +1, +2\}$ is the domain of value variations for y_2.

Now, let us consider that, in the current problem, $x_1^0 = 3$ (*Analyzer Type* in source problem), $x_1^1 = 2$ (*Analyzer Type* in target problem), and $y_2^0 = 3$ (*Moderation Level* in source solution). Then the value of y_2^1 (*Moderation Level* in target solution) is computed as follows:

$d_1y_2 = \frac{\partial y_2}{\partial x_1}(x^0) \times dx_1$ where $dx_1 = x_1^1 - x_1^0 = 2 - 3 = -1$ and -1 in ΔV_1 corresponds to -1 in ΔW_2, thus $\frac{\partial y_2}{\partial x_1}(x^0) = 1/1 = 1$ and $d_1y_2 = \frac{\partial y_2}{\partial x_1}(x^0) \times dx_1 = -1$.

The final adapted value for ***Moderation Level*** (y_2) is:

$$y_2^1 = y_2^0 + d_1y_2 + d_7y_2 = 3 + (-1) + (+2) = 4.$$

A difference of 1 on ***Analysis Class*** (x_7) entails a higher value for ***Moderation Level*** and a more moderated program, while a difference of 1 on ***Analyzer Type*** (x_1) entails a smaller value for ***Moderation Level*** and thus a less moderated program. This is a simplified but effective application of differential adaptation in a real-world context.

Below, we conclude this section with a brief comment on the history and the use of the Prolabo application in industry. The Prolabo system was initiated around 1995 and was presented for the first time during a workshop on CBR Adaptation at ECAI-96 [28]. Based on a first prototype, a system was developed for guiding the search for digestion programs with the help of keywords, e.g. title and description of programs, categories of problems such as air, food, soil, water... One option in the system was based on CBR and the case base was including digestion plans minimizing the time of the process, energy consumption, number of steps and chemical products to be used. The digestion devices were sold to specialized laboratories which were also able to extend the case base with their own experiences. Some of these new cases could be integrated in Prolabo as soon as their interest could be recognized.

To conclude this section, we would like to mention how the Prolabo system was appraised when it was in activity. Unfortunately, there does not exist any formal feedback report related to the use of the Prolabo system. The Prolabo system was appreciated by the technical staff of the laboratories where the system was working: it appeared that the digestion process was five times quicker when guided by Prolabo and that the reuse of cases was more easy and informative for engineers. The system was still in use at the beginning of the 2000s, but then the Prolabo society was merged with the BDH society[2]

[2] `https://fr.vwr.com/app/Header?tmpl=/vwr_collection/bdh_prolabo.htm`

and the industrial objectives of the resulting consortium changed and selling digestion program is no more a priority of the new society.

4. Discussion

The main characteristics of differential adaptation can be summarized as follows:

- Retrieval and adaptation are intertwined: according to the adaptation-guided retrieval principle [37], a source case is considered as similar if it is possible to adapt its solution to the target case. The adaptation process can be performed gradually by building an adaptation path –in the solution space– aligned with a dual similarity path –in the problem space.

- Adaptation knowledge represents dependencies between problem descriptors and solution descriptors: there is a dependency if and only if a variation of the value of a problem descriptor entails a variation of (at least) a solution descriptor.

- Adaptation knowledge represents the influence of a variation of a problem descriptor on the variation of a solution descriptor.

Moreover, let us mention that interactions are not taken into account within differential adaptation. Interactions may occur when an adaptation operation interacts with another adaptation operation, preventing the achievement of the solution. Interactions can be avoided in some situations when they can be detected and then modifying and introducing appropriate descriptors and dependencies. Further studies have to be carried out to examine this problem more carefully.

Finally, based on the preceding section, two important issues on knowledge engineering for case-based reasoning can be noticed:

- Case elaboration –or *what's in a case*– needs a large amount of knowledge on the way of adapting a case; connections can be made with the objectives of knowledge intensive CBR [1, 30, 36]. Actually, the representation of a case for being manipulated by CBR is probably different from the user representation. In particular, the Prolabo experience shows that the representation of a case must be well-known by the user of the case-based reasoning system.

- Influences constitute the main knowledge units for adaptation and for similarity assessment. Influences do not generally rely on 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 differences to be adjusted between problem descriptors. Knowledge on influences is of first importance and should be elicited and/or mined (see for example [16] or [15] where they are obtained through a knowledge discovery process).

Differential adaptation has some limitations and the first one is to be well adapted to numerical problems, because of the metaphor of differential calculus, but much less actionable in other more symbolic domains. Actually, the availability of adaptation knowledge is a major concern for designing a good differential adaptation process. Moreover, we have seen in the Prolabo application that it is important to know the interactions between descriptors, in problems and solutions, and to be able to quantify these variations w.r.t. domains of values of the descriptors. Actually, the lack of a complete description of the problem descriptors and their variations prevents the application of differential adaptation. Thus, problems that can be represented or approximated by mathematical functions are good candidates for being solved by a differential adaptation approach. Indeed, this is confirmed by the experiments which are proposed in the next section.

5. Experiments

We have conducted some experiments to show the behavior, the usefulness and the efficiency of the differential adaptation approach. In this section, we present and discuss the results of these experiments. The global setting of the experiments relies on the use of numerical functions that are known but that should be approximated based on the differential adaptation framework. More precisely, we have the following setting:

- a domain $D = [a_1, b_1] \times \ldots \times [a_m, b_m] \subseteq \mathbb{R}^m$ where m is a non null integer,
- a (target) function $f : D \longrightarrow \mathbb{R}^n$ which is continuous and differentiable, where n is a non null integer,

- the matrix of partial derivatives $f : J_f = \left[\frac{\partial dy_j}{\partial dx_i}\right]_{ij}$ is used as domain knowledge for differential adaptation.

A source case is a pair (x, y) where $x \in \mathbb{R}^m$ denotes a problem and $y \in \mathbb{R}^n$ denotes a solution. The target problem x can be randomly chosen on D and $y = f(x)$ is computed accordingly. The case base can also be randomly defined. Case retrieval is based on the Euclidean distance on $\mathbb{R}^m$. Then the closest case w.r.t. Euclidean distance is selected.

$$d(x^0, x^q) = \sqrt{\sum_{i=1}^{m}(x_i^q - x_i^0)}$$

The experiment consists in comparing the approximation of the target function using on one side differential adaptation and on the other side "adaptation by copy" or "null adaptation". In both approaches, the selected case is closest case w.r.t. Euclidean distance. In adaptation by copy, the selected cased is reused as such. In differential adaptation, the selected case is transformed following 4 variations corresponding to the length of the similarity path: "basic differential adaptation" with $q = 1$, i.e. no intermediate problem, $q = 2$ with one intermediate problems, $q = 3$ with two intermediate problems, and finally $q = 10$ with nine intermediate problems. By contrast, adaptation by copy (also known as *null adaptation*) leads to a "retrieval-only CBR system".

For a given target problem x^0 and a case base B we compute:

- a series of intermediate problems: $x^i = x^0 + \frac{i}{q}(x^q - x^0)$, $i \in \{1, \ldots, q-1\}$,
- the error induced by basic differential adaptation ($q = 1$):
  $e_{ad}(B, x^0) = |f(x^0) - ad(B, x^0)|$, where $ad(B, x^0) \in \mathbb{R}^n$ denotes the results computed by the CBR system with differential adaptation,
- the error induced by differential adaptation for $q \in \{2, 3, 10\}$,
- the error induced by adaptation by copy:
  $e_{apc}(B, x^0) = |f(x^0) - apc(B, x^0)|$, where $apc(B, x^0) \in \mathbb{R}^n$ denotes the results computed by the CBR system with adaptation by copy.

Two main experiments were conducted with the following settings:

FIRST EXPERIMENT.

First experiment, size of the case base: 10

Average error DA = 0.535983896644	standard deviation = 0.631967899962
Average error DA q2 = 0.237122064473	standard deviation = 0.257023208132
Average error DA q3 = 0.156531637355	standard deviation = 0.167246531346
Average error DA q10 = 0.0467083293785	standard deviation = 0.0492802200672
Average error AbC = 0.464678852451	standard deviation = 0.355187010861

First experiment, size of the case base: 100

Average error DA = 0.0579929164291	standard deviation = 0.0737863767981
Average error DA q2 = 0.0286177653139	standard deviation = 0.0365515950717
Average error DA q3 = 0.0190908362735	standard deviation = 0.0244415410557
Average error DA q10 = 0.00575092345918	standard deviation = 0.00739476177453
Average error AbC = 0.176773258534	standard deviation = 0.153918289853

First experiment, size of the case base: 1000

Average error DA = 0.00569897202888	standard deviation = 0.00721542323048
Average error DA q2 = 0.00284594850936	standard deviation = 0.00362319210742
Average error DA q3 = 0.00189679834486	standard deviation = 0.00241964095184
Average error DA q10 = 0.000568829613377	standard deviation = 0.000727892191937
Average error AbC = 0.0547685311177	standard deviation = 0.0478299583741

SECOND EXPERIMENT.

Second experiment, size of the case base: 10

Average error DA = 0.0496651787182	standard deviation = 0.140085951689
Average error DA q2 = 0.0233913678102	standard deviation = 0.06150697174
Average error DA q3 = 0.0153527418068	standard deviation = 0.0396141924762
Average error DA q10 = 0.00452215160882	standard deviation = 0.0113927071569
Average error AbC = 0.139138776939	standard deviation = 0.214324533687

Second experiment, size of the case base: 100

Average error DA = 0.000528066579352	standard deviation = 0.00140273310711
Average error DA q2 = 0.000264721065554	standard deviation = 0.0007025558266
Average error DA q3 = 0.000176634928872	standard deviation = 0.000468666554057
Average error DA q10 = 5.30562251765e-05	standard deviation = 0.000140732092041
Average error APC = 0.0131955221431	standard deviation = 0.0205973661982

Second experiment, size of the case base: 1000

Average error DA = 5.75013575637e-06	standard deviation = 1.74352570251e-05
Average error DA q2 = 2.87508632519e-06	standard deviation = 8.71421440129e-06
Average error DA q3 = 1.91672831761e-06	standard deviation = 5.8087218258e-06
Average error DA q10 = 5.75020217813e-07	standard deviation = 1.74230075132e-06
Average error AbC = 0.00139535789666	standard deviation = 0.00219893043654

Figure 5: The numerical results of the experiments.

- $m = 2$, $n = 1$, $D = [-4., 4.] \times [-4., 4.]$,
  $f(x) = \sin x_1 \cos x_2$, $J_f = \begin{bmatrix} \cos x_1 \cos x_2 \\ -\sin x_1 \sin x_2 \end{bmatrix}$
- $m = 1$ ($x_1 = x$), $n = 1$, $D = [-4., 4.]$, $f(x) = \dfrac{\sin x}{1 + e^x}$,
  $J_f = \left[\dfrac{\cos x(1 + e^x) - \sin x \cdot e^x}{(1 + e^x)^2} \right]$

The case base has 3 different sizes, namely 10, 100, 1000 cases. The process hereafter is then repeated 1000 times: (i) random generation of the case base, (ii) random generation of the target problem, (iii) computing the

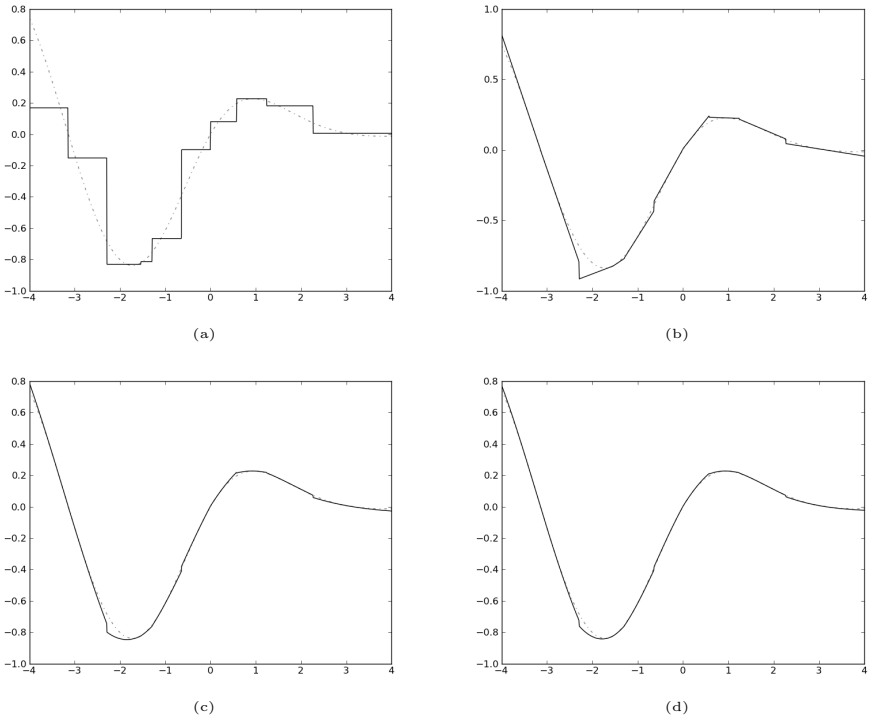


Figure 6: Approximation of the target function: experiments for 10 cases.

average error, (iv) computing the standard deviation. The results are given in Figure 5.

Moreover, Figure 6 and Figure 7 present four curves each time with a case base B respectively including 10 cases and 100 cases which are randomly generated. In each figure, the dotted line corresponds to the target function $f(x) = \dfrac{\sin x}{1 + e^x}$. In (a), the other curve corresponds to the result of adaptation by copy. In (b) (respectively in (c), (d)), the other curve corresponds to the result of differential with $q = 1$ (respectively $q = 2$, $q = 3$).

Then we can observe the following phenomena:

- Looking at (a) and (b) in Figure 6 or in Figure 7, one can observe that the quality of differential adaptation with $q = 1$ is much better than that of adaptation by copy. This is the first main observation.

- The influence of parameter q: looking at curves (b), (c), and (d) in

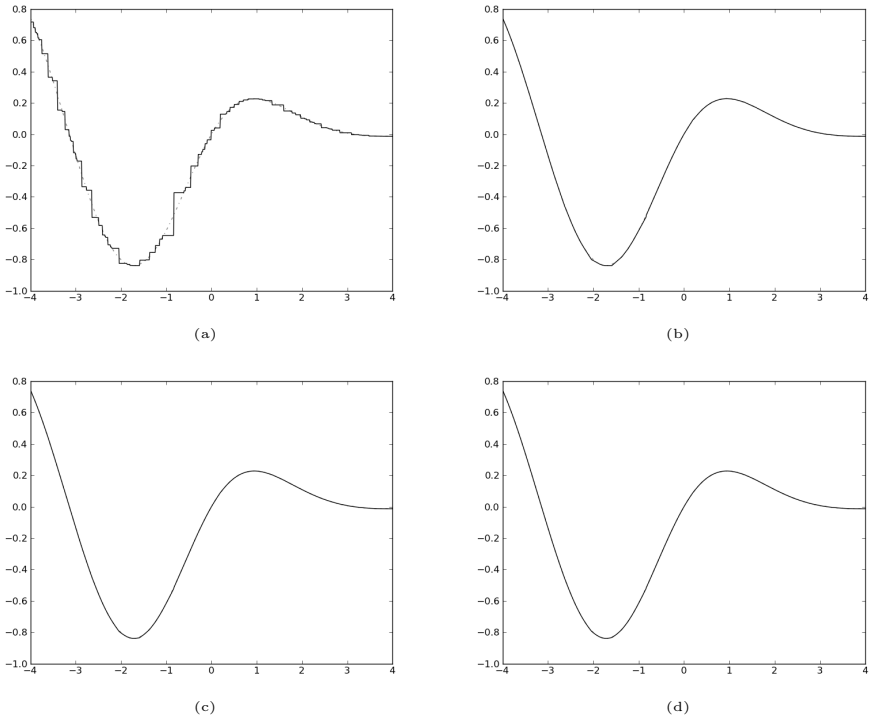


Figure 7: Approximation of the target function: experiments for 100 cases.

Figure 6 and Figure 7, we can observe that the more q is high the better differential adaptation is, even if it is more easy to check this fact in Figure 6. This is the second main observation.

- The influence of the size of the case base: another time, looking at curves (b), (c), and (d) in both Figure 6 and Figure 7, we can observe that the approximation is of higher quality in Figure 7 with 100 cases than in than in Figure 6 with 10 cases. This is the third main observation of these experiments.

This interpretation concludes this section on experiments and shows the very good behavior of differential adaptation in solving the problem of mathematical function approximation with a CBR approach. This shows also that being interested in *dependencies between descriptors* in adaptation is indeed a good idea and can lead to good implementations of the adaptation process.

6. Related work

In this section, we discuss the place of differential adaptation w.r.t. close approaches. It can be noticed that references are not so recent as it seems that adaptation did not deserve so much attention in the CBR literature, at least for works close to differential adaptation. However, we will conclude the section with recent work on revision-based adaptation which is a complementary approach to differential 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 and for 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 [4, 3], dealing with case abstraction and planning problem-solving.

In [31], the formalization of adaptation in the context of design problems is based on a particular case representation allowing case processing with CSP (Constraint Satisfaction Problems) methods. Following a similar approach as in [4, 3], cases are decomposed into sub-cases, and global consistency is guaranteed by the constraint-solving method, which relies on an efficient heuristic: the source cases minimizing the number of constraints to be solved have to be preferred. Dependencies are expressed as constraints to be satisfied and the solver needs a detailed domain model for completing an efficient processing.

The two preceding approaches are mainly related to generative case adaptation, while some research papers have addressed transformational and substitution adaptation processes. In [5], 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 appropriate 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 [8], 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 types of the descriptors.

In [2], 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. Based on similar ideas, substitutional adaptation is provided with a working implementation for solving an industrial problem in [26].

In [20], 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. More recently, this research work has been extended in the context of knowledge intensive CBR [36]. The authors propose a domain independent algorithm for case adaptation based on planning techniques and heavily depending on a domain ontology. Here again, the role and the use of domain knowledge is considered as being of high importance.

Among more recent work on adaptation, [29] presents case-based adaptation of workflows. Adaptation is based on an anchor mapping algorithm which identifies the parts of the workflow where to apply the changes. In [22], a method for increasing the context-awareness of case adaptation is proposed, with an application to a regression task. In this work, the context is used to improve the quality of adaptation rules that can be automatically extracted from the comparison of the cases. In [33], adaptation in case-based design relies on k-nearest neighbors and is used for feature adaptation. These three papers show recent research directions on case-based reasoning and adaptation, and propose related adaptation strategies. However, they are not closely related to differential adaptation which relies on a different basis, namely differential calculus, which provides the rules for "computing adaptation" with the help of similarity paths and domain knowledge. Nevertheless, all approaches show that domain knowledge should be present for conducting a successful adaptation process.

For ending this discussion and providing a comparison with related work, we first summarize the characteristics of differential adaptation and then we compare these characteristics with respect to related work (see Table 3).

A1. Retrieval and adaptation are intertwined: a source case is selected in the case base if and only if it is possible to adapt its solution for solving the target problem.

A2. Adaptation knowledge represents dependencies between problem de-

Approach to adaptation	A1	A2	A3	A4	A5	A6
Differential Adaptation, transformational adaptation [our approach]	x	x	x	x	x	x
Constraints guided adaptation; generative case adaptation [6, 3, 4, 31]	x	x				
Local adaptation operators, global adaptation function, transformational and substitutional adaptation [5]	x	x		x		x
Adaptation by interpolation, transformational adaptation [8]	x	x	x	x		
Adaptation by interpolation, using a common metric problem-solution transformational adaptation [2]	x	x	x			
Substitutional adaptation, memory search strategy [20]	x	x				x

Table 3: A comparison of differential adaptation with related adaptation approaches.

scriptors and solution descriptors of a case: there is a dependency if and only if a variation of the value of a problem descriptor entails a variation of (at least) a solution descriptor.

A3. Adaptation knowledge represents the influence of a variation of a problem descriptor on the variation of a solution descriptor: this influence is expressed in terms of a proportion for being connected with a similarity measure.

A4. The adaptation process uses specific adaptation operators taking into account a difference of value on problem descriptors for computing a difference of value on dependent solution descriptors.

A5. The adaptation process can be performed gradually by building an adaptation path (in the solution space) aligned with a dual similarity path (in the problem space).

A6. There exists a general algorithm for adaptation and this algorithm uses adaptation knowledge.

We conclude this section by giving a brief presentation of *revision-based adaptation* which can be considered as a complementary approach to differential adaptation, especially studied in logical formalisms [9, 10, 17]. Given a set of beliefs ψ and new beliefs μ, the revision of ψ by μ consists in minimally modifying ψ into ψ' such that it is consistent with μ. Then, the revision of ψ by μ is $\psi \dot{+} \mu = \psi' \wedge \mu$. Given a distance function, say *dist* on the set of interpretations of the formulas in the given formalism, a revision operator $\dot{+}^{dist}$ is defined. The models of $\psi \dot{+}^{dist} \mu$ are the models of μ that are the closest ones to the models of ψ, according to *dist*.

Revision-based adaptation can be defined as follows. The solution $\texttt{Sol}(\texttt{tgt})$ of the target problem $\texttt{tgt}$ is such that:

$$\texttt{DK} \wedge \texttt{tgt} \wedge \texttt{sol}(\texttt{tgt}) \equiv (\texttt{DK} \wedge \texttt{srce} \wedge \texttt{sol}(\texttt{srce})) \dot{+} (\texttt{DK} \wedge \texttt{tgt})$$

where $(\texttt{srce}, \texttt{Sol}(\texttt{srce}))$ is the case to be adapted and $\texttt{DK}$ refers to domain knowledge, i.e. the available knowledge shared by all cases. Intuitively: the source case is minimally modified, according to $\dot{+}$, in order to reach consistency with the target problem, taking into consideration domain knowledge.

Revision-based adaptation has been investigated in several formalisms (see [11] for a synthesis) and is useful to specify and implement adaptation once an appropriate notion of "minimal modification" is modeled (thanks to a distance function). By contrast, differential adaptation is useful to to specify a revision operator and to investigate how the notion of minimal modification can be modeled. Finally, it is worth to mention that domain knowledge can be explicitly introduced and used in revision-based adaptation.

7. Conclusion and future work

In this paper, we have presented differential adaptation, a framework for solving numerical problems with CBR. Adaptation is considered as a central step in CBR, 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. Differential adaptation takes into account these two dimensions for building a solution of the target problem.

The adaptation process is decomposed to yield a sequence of intermediate problems between the source and target problems (similarity path). Local operators have the ability to solve the intermediate problems. Then, based on the metaphor of differential calculus, global operators merge the local solutions for building a global solution to the target problem. Differential adaptation proposes an operational framework and is used in a real-world context.

Going further, CBR needs 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. When knowledge engineers are implementing knowledge systems aimed at solving real-world problems, they should consider CBR as a powerful inference schema, completing deduction and induction schemes. Following this idea, a complete knowledge system should rely on a knowledge base, on a case base, and in addition should be assisted by a knowledge discovery system.

References

[1] A. Aamodt. Knowledge-intensive case-based reasoning and sustained learning. In L. Aiello, editor, *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI-90)*, pages 1–6. Pitman, London/Boston, 1990.

[2] P. Avesani and E. Blanzieri. Adaptation-dependent retrieval problem: A formal definition. In J. Lieber, E. Melis, A. Mille, and A. Napoli, editors, *Workshop on the Formalisation of Adaptation in Case-Based Reasoning at ICCBR'99, Seeon Monastery, Germany*. University of Kaiserslautern (S. Schmitt and I. Vollrath volume editors), 1999.

[3] R. Bergmann and W. Wilke. Building and Refining Abstract Planning Cases by Change of Representation Language. *Journal of Artificial Intelligence Research*, 3:53–118, 1995.

[4] R. Bergmann and W. 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 at ECAI-96 Conference, Budapest, Hungary*. ECCAI/NJSZT Budapest, 1996.

[5] R. Bergmann and W. Wilke. Towards a New Formal Model of Transformational Adaptation in Case-Based Reasoning. In H. Prade, editor, *Proceedings of ECAI 98*, pages 53–57. John Wiley and Sons, Ltd, 1998.

[6] R. Bergmann, G. Pews, and W. Wilke. Explanation-based similarity: a unifying approach for integrating domain knowledge into case-based reasoning for diagnosis and planning tasks. In S. Wess, K.-D. Althoff, and M.M. Richter, editors, *Topics in Case-Based Reasoning – First European Workshop (EWCBR'93)*, Lecture Notes in Artificial Intelligence 837, pages 182–196. Springer, 1994.

[7] R. Bergmann, J.L. Kolodner, and E. Plaza. Representation in case-based reasoning. *Knowledge Engineering Review*, 20(3):209–213, 2005.

[8] N. Chatterjee and J.A. Campbell. Interpolation as a means of fast adaptation in case-based problem solving. In R. Bergmann and W. Wilke, editors, *Fifth German Workshop on Case-Based Reasoning*, pages 65–74. Technical Report LSA-97-01E, University of Kaiserslautern, Germany, 1997.

[9] J. Cojan and J. Lieber. An algorithm for adapting cases represented in an expressive description logic. In I. Bichindaritz and S. Montani, editors, *ICCBR*, Lecture Notes in Computer Science 6176, pages 51–65. Springer, 2010.

[10] J. Cojan and J. Lieber. An algorithm for adapting cases represented in alc. In T. Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona (IJCAI 2011)*, pages 2582–2589. IJCAI/AAAI, 2011.

[11] J. Cojan and J. Lieber. Applying belief revision to case-based reasoning. In H. Prade and G. Richard, editors, *Computational Approaches to Analogical Reasoning: Current Trends*, volume 548 of *Studies in Computational Intelligence*. Springer, 2014.

[12] A. Cordier, B. Fuchs, and A. Mille. Engineering and Learning of Adaptation Knowledge in Case-Based Reasoning. In *Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW-2006), Podebrady, Czech Republic*, Lecture notes in Artificial Intelligence 4248, pages 303–317. Springer, 2006.

[13] A. Cordier, B. Fuchs, L. Lana de Carvalho, J. Lieber, and A. Mille. Opportunistic acquisition of adaptation knowledge and cases - the iaka approach. In K.-D. Althoff, R. Bergmann, M. Minor, and A. Hanft, editors, *ECCBR*, Lecture Notes in Computer Science 5239, pages 150–164. Springer, 2008.

[14] R. Courant and F. John. *Introduction to Calculus and Analysis (Volumes 1 and 2)*. Springer, 2000.

[15] S. Craw, N. Wiratunga, and R.C. Rowe. Learning adaptation knowledge to improve case-based reasoning. *Artificial Intelligence*, 170(16–17):1175–1192, 2006.

[16] M. d'Aquin, F. Badra, S. Lafrogne, J. Lieber, A. Napoli, and L. Szathmary. Case base mining for adaptation knowledge acquisition. In M.M. Veloso, editor, *Proceedings of IJCAI 2007*, pages 750–755. Morgan Kaufman, 2007.

[17] V. Dufour-Lussier, F. Le Ber, J. Lieber, and L. Martin. Case adaptation with qualitative algebras. In F. Rossi, editor, *IJCAI*, pages 3002–3006. AAAI Press, 2013.

[18] B. Fuchs, J. Lieber, A. Mille, and A. Napoli. Towards a unified theory of adaption in case-based reasoning. In K. Branting, K.-D. Althoff, and R. Bergmann, editors, *Proceedings of the Third International Conference on Case-Based Reasoning, ICCBR-99*, Lecture Notes in Articificial Intelligence 1650, pages 104–117. Springer, 1999.

[19] B. Fuchs, J. Lieber, A. Mille, and A. Napoli. An Algorithm for Adaptation in Case-based Reasoning. In W. Horn, editor, *Proceedings of ECAI-2000*, pages 45–49. IOS Press, Amsterdam, 2000.

[20] P.A. González-Calero, M. Gómez-Albarrán, and B. Díaz-Agudo. A substitution-based adaptation model. In S. Schmitt and I. Vollrath, editors, *Challenges for Case-Based Reasoning - Proceedings of the ICCBR'99 Workshops, Seeon Monastery, Germany, July 27-30, 1999*, pages 17–26. University of Kaiserslautern (Computer Science), 1999.

[21] S. Hanks and D.S. Weld. A Domain-Independent Algorithm for Plan Adaptation. *Journal of Artificial Intelligence Research*, 2:319–360, 1995.

[22] V. Jalali and D. Leake. A context-aware approach to selecting adaptations for case-based reasoning. In P. Brézillon, P. Blackburn, and R. Dapoigny, editors, *CONTEXT*, Lecture Notes in Computer Science 8175, pages 101–114. Springer, 2013.

[23] J.L. Kolodner. *Case-Based Reasoning*. Morgan Kaufman Publishers, San Mateo, CA, 1993.

[24] J. Lieber and A. Napoli. Using Classification in Case-Based Planning. In W. Wahlster, editor, *European Conference on Artificial Intelligence (ECAI'96)*, pages 132–136. John Wiley & Sons Ltd, Chichester, 1996.

[25] R. Lopez De Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M.L. Maher, M.T. Cox, K. Forbus, M. Keane, A. Aamodt, and I. Watson. Retrieval, reuse, revision and retention in case-based reasoning. *Knowledge Engineering Review*, 20(3):215–240, 2005.

[26] S. Manzoni, F. Sartori, and G. Vizzari. Substitutional Adaptation in Case-Based Reasoning: A General Framework Applied to P-Truck Curing. *Applied Artificial Intelligence*, 21(4&5):427–442, 2007.

[27] E. Melis, J. Lieber, and A. 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.

[28] A. Mille, B. Fuchs, and O. Herbeaux. A unifying framework for adaptation in case-based reasoning. In A. Voss, R. Bergmann, and B. Bartsch-Spörl, editors, *Workshop on Adaptation in Case-Based Reasoning at ECAI-96 Conference, Budapest, Hungary*, pages 22–28. ECCAI/NJSZT Budapest, 1996.

[29] M. Minor, R. Bergmann, and S. Görg. Case-based adaptation of workflows. *Information Systems*, 40:142–152, 2014.

[30] P. Öztürk and A. Aamodt. A context model for knowledge-intensive case-based reasoning. *International Journal of Human Computer Studies*, 48(3):331–355, 1998.

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

[32] L. Purvis and P. Pu. Adaptation using constraint satisfaction techniques. In M. Veloso and A. Aamodt, editors, *Proceedings of the First International Conference on Case-Based Reasoning (ICCBR'95)*, Lecture Notes in Artificial Intelligence 1010, pages 289–300, Berlin, 1995. Springer.

[33] J. Qi, J. Hu, and Y.-H. Peng. A new adaptation method based on adaptability under k-nearest neighbors for case adaptation in case-based design. *Expert Systems Applications*, 39(7):6485–6502, 2012.

[34] M. Richter. Introduction. In M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, and S. Wess, editors, *Case-Based Reasoning Technology*, Lecture Notes in Computer Science 1400, pages 1–16. Springer, 1998.

[35] C.K. Riesbeck and R.C. Schank. *Inside Case-Based Reasoning.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1989.

[36] A.A. Sánchez-Ruiz-Granados, P.P. Gómez-Martín, B. Díaz-Agudo, and P.A. González-Calero. Adaptation through planning in knowledge intensive cbr. In K.-D. Althoff, R. Bergmann, M. Minor, and A. Hanft, editors, *Advances in Case-Based Reasoning, Proceedings of the 9th European Conference (ECCBR 2008)*, Lecture Notes in Computer Science 5239, pages 503–517. Springer, 2008.

[37] B. Smyth and M. T. Keane. Using adaptation knowledge to retrieve and adapt design cases. *Knowledge-Based Systems*, 9(2):127–135, 1996.

[38] A. Voss. Case reusing systems – survey, framework and guidelines. *The Knowledge Engineering Review*, 12(1):59–89, 1997.