Multi-modal Diagnosis Combining Case-Based and Model-Based Reasoning: a Formal and Experimental Analysis

Luigi Portinale
Dipartimento di Informatica
Universitá del Piemonte Orientale "Amedeo Avogadro"
e-mail: portinal@unipmn.it

Diego Magro, Pietro Torasso
Dipartimento di Informatica
Universitá di Torino
e-mail: {magro,torasso}@di.unito.it

TR-INF-2003-12-08-unipmn

Keywords: multi-modal reasoning, case-based reasoning, opportunistic reasoning, case-based integration.

Corresponding Author:
Luigi Portinale
Dipartimento di Informatica - Universitá del Piemonte Orientale "A. Avogadro"
Spalto marengo 33 - 15100 Alessandria (ITALY)
tel: +39 0131 287 451 fax: +39 0131 287 440
e-mail: portinal@unipmn.it

Abstract

Integrating different reasoning modes in the construction of an intelligent system is one of the most interesting and challenging aspects of modern AI. Exploiting the complementarity and the synergy of different approaches is one of the main motivations that led several researchers to investigate the possibilities of building *multi-modal reasoning systems*, where different reasoning modalities and different knowledge representation formalisms are integrated and combined. Case-Based Reasoning (CBR) is often considered a fundamental modality in several multi-modal reasoning systems; CBR integration has been shown very useful and practical in several domains and tasks. The right way of devising a CBR integration is however very complex and a principled way of combining different modalities is needed, in order to gain the maximum of efficacy and efficiency for a particular task. In the present paper we present results (both theoretical and experimental) concerning architectures integrating CBR and Model-Based Reasoning (MBR) in the context of diagnostic problem solving. We first show that both an MBR approach to diagnosis and a CBR one may suffer from computational intractability, and therefore a careful combination of the two approaches may be useful to reduce the computational cost in the average case. A main contribution of the paper concerns the fact that we analyze the different facets that may influence the whole performance of a multi-modal reasoning system, namely computational complexity, system competence in problem solving and the quality of the sets of produced solutions. We show that an opportunistic and flexible architecture able to estimate the right cooperation among modalities can exhibit a satisfactory behavior with respect to every performance aspect. An analysis of different ways of integrating CBR is performed both at the experimental and at the analytical level. On the analytical side, a cost model and a competence model able to analyze a multi-modal architecture through the analysis of its individual components are introduced and discussed. On the experimental side, a very detailed set of experiments has been carried out, showing that a flexible and opportunistic integration can provide significant advantages in the use of a multi-modal architecture.

1 Introduction

The difficulty of representing in a single formalism all the kinds of knowledge about a complex system (e.g. an artifact) and of defining efficient reasoning mechanisms for solving problems in demanding tasks such as diagnosis and planning are well know problems. These problems have received a significant amount of attention in the past and there has been a recent impulse in the study of *multi-modal reasoning systems*, because of the importance of devising flexible architectures able to integrate and combine different reasoning modalities [3, 19, 49]. The main motivation for integrating different reasoning methods and styles is to exploit complementarities and achieve a synergy which produce results that could not be obtained by using each reasoning mode individually [43]. A reasoning modality that is considered fundamental in several multi-modal systems is certainly Case-Based Reasoning (CBR) [23, 1, 17]. The design of a multi-modal architecture very often provides reasoning components able to perform CBR in a combined way with other reasoning modes [3, 19, 43, 2]; this is so evident in both theoretical research and applications that the name *CBR integrations* is now currently used to identify an explicit research subfield of CBR [29]. A number of possible CBR integrations has been investigated considering almost every kind of automated reasoning paradigm, ranging from rule-based reasoning, to model-based reasoning, constraint satisfaction, soft computing etc...

However, the experience gained in developing such a kind of multi-modal reasoning systems pointed out that approaches based on multiple representations require the solutions to not easy problems; among them, the selection of specific representations which can be actually useful for the task at hand and the way different representations are used (and when) by a problem solver, in order to gain the maximum of efficacy and efficiency for a particular task [19]. As pointed out in [29], these problems reflect in two open issues concerning CBR integrations:

- determining to what extent different reasoning modalities need different knowledge representations and how such representations are integrated;
- finding the best system architecture for the integrated system.

In the present paper we aim at addressing the above issues, by taking into consideration a specific kind of CBR integration, namely the combination of CBR with Model-Based Reasoning (MBR) applied to diagnostic tasks. In particular, we propose a principled way of integrating CBR and MBR for diagnosis, based on both a theoretical and experimental analysis. An important reason supporting the approach is that a principled integration can provide significant performance advantages. The inherent complexity of the reasoning mechanisms involved in first principle systems makes their problem solving activity very heavy from a computational point of view if real-world domains are taken into consideration [12, 40]. For this reason a number of approaches for speeding-up problem solving have been proposed and the benefits of supplementing a first-principle problem solver (theoretically able to solve any solvable problem in the modeled domain) with a case-based component were made apparent by the experimental analysis of some of the early systems complementing reasoning from first-principle with CBR (e.g. CASEY [24], KRITIK [20], PRODIGY-ANALOGY [50]).

On the other hand, an in-depth theoretical analysis of the reasoning mechanisms involved in re-using past solution shows that there is not always the guarantee of improving problem solving from the computational point of view. The seminal work by Nebel and Koehler on case-based planning showed that the theoretical computational complexity of re-using plans

of previously solved problems is at least as hard as generating the plans from scratch, when a conservative approach is taken [31]. The problem is not peculiar to planning: in [33], we carried on a theoretical analysis for diagnostic problem solving by comparing a model-based approach (where a problem is solved from scratch) with an approach involving re-use of the past solutions, showing results very similar to those obtained in [31]. Then it becomes important to investigate under which conditions a multi-modal approach combining a case-based component with a problem solver from first-principle provides better performance than just a first-principle problem solver alone.

The main goal of the paper is to show that a principled way for integrating a case-based reasoning component in a multi-modal architecture is needed, in order to take into account all the performance facets of the global architecture. In particular we emphasize that there is no single parameter able to capture the different aspects of the system performance. In fact, one could consider the ability of the system to solve different types of problems, the resources it needs for solving the problem (both in term of time and space) and the quality of the solutions it produces. Any kind of mechanism for improving a performance facet has to take into account the *utility problem* [30]; this is the arising of performance degradation in a knowledge-based system when knowledge is added in a undistinguished way to its knowledge base, without considering the actual usefulness of such a knowledge (i.e. searching for the right chunk of knowledge to use may override the benefits of having more knowledge available). The utility problem affects most approaches, having in particular a strong impact on CBR (where it is called the *swamping problem*) [18, 41] and on multi-modal architectures involving CBR [34, 38, 48].

In the present paper we present results (both theoretical and experimental) concerning architectures integrating Case-Based Reasoning (CBR) and Model-Based Reasoning (MBR) in the context of diagnostic problem solving. In particular, we stress the following points:

- different modalities (CBR and MBR in our case) must share some common knowledge representation mechanism, in order to successfully cooperate: in our case the model used to perform MBR is also used by the CBR component for every task involving adaptation (namely adaptation-guided retrieval and solution adaptation itself);

- an opportunistic and flexible architecture able to estimate the right modality or cooperation among modalities can provide significant advantages with respect to every performance facet.

As concerns theory, having characterized in a formal way the notion of model-based diagnosis in Section 2, in Section 3 we summarize results on the computational complexity of solving diagnostic problem by adopting a pure MBR approach with respect to the computational complexity of re-using and adapting solutions of previously solved problems. In Section 4, we sketch the basic architecture of **ADAPtER**, a diagnostic system based on a multi-modal reasoning approach, combining CBR and MBR in such a way that the basic reasoning mechanisms used by the MBR component for solving a problem from scratch, can be used in a focused way by the CBR component to implement the classical adaptation step. This realizes a *master/slave integration* [29] where CBR is the main reasoning component and MBR is essentially used to fill competence gaps of the case-base and to guide the adaptation step of the CBR cycle.

The analysis of this kind of integration, with respect to the various performance dimensions related to the utility problem, is then carried out both at the experimental level (Section5) and at the analytical level (Section 6); in particular, a cost and competence model able to analyse

a multi-modal architecture through the analysis of its individual components is introduced and discussed. Such analyses point out the needs for more flexible ways of integrating the reasoning modes, by suggesting the use of more *opportunistic* strategies [22, 17] able to choose the more suitable reasoning mode depending on the type of problem to be solved. Section 7 discusses this opportunistic integration in details and shows the advantages of this kind of integration. Finally some conclusions and a comparison with other multi-modal proposals found in the literature are reported in Section 8.

2 Characterizing Diagnostic Problems

In order to discuss both theoretical aspects of adaptation complexity and practical implementation, we introduce the formal framework we refer to for characterizing diagnostic problems. The framework has been initially proposed in [14] as a general approach able to unify classical approaches to model-based diagnosis (namely purely consistency-based and purely abductive diagnosis).

Definition 2.1 *A diagnostic problem is a tuple* $DP = \langle BM, HYP, CXT, \langle \Psi^+, \Psi^- \rangle \rangle$ *where:*

- BM *is a set of definite clauses (without recursion) representing the behavioral model of the system to be diagnosed. In particular, the clauses should have non-empty bodies;*
- HYP *is a finite set of ground atoms of* BM*, whose predicates are called* abducibles*, representing possible diagnostic hypotheses. Abducibles can appear only in the body of clauses;*
- CXT *is a finite set of ground atoms of* BM *representing contextual information characterizing the diagnostic problem. Context atoms can appear only in the body of clauses and always in conjunction with some atom not belonging to* CXT*;*
- Ψ^+ *is a finite set of ground atoms of* BM *representing the observations to be accounted for (i.e. covered) in the current case;*
- Ψ^- *is a finite set of ground atoms of* BM *representing the instances of observable parameters that conflict with the observations.*

In the above definition, we characterize a diagnostic problem in terms of Ψ^+ and Ψ^-. Actually, a diagnostic problem is characterized by OBS, the set of observations available for the problem under examination. While Ψ^- is uniquely determined given OBS according to the criterion $\Psi^- = \{m(y) | m(x) \in OBS \wedge x \neq y\}$, there are many possible ways for determining Ψ^+ from OBS, since we only impose that $\Psi^+ \subseteq OBS$ [1].

We assume that each predicate occurring in BM has a finite set of ground instances[2]. Moreover, since we abstract from temporal aspects we also assume that the following meta-level constraint holds for every predicate symbol p:

$$p(x) \wedge p(y) \rightarrow \bot \quad (x \neq y)$$

[1] Different choices of Ψ^+ give raise to quite different definitions of diagnosis (see [14]), ranging from purely abductive definitions ($\Psi^+ = OBS$) to purely consistency-based definitions ($\Psi^+ = \emptyset$).

[2] Therefore, BM is equivalent to a propositional definite clause theory.

A set of ground atoms is consistent if and only if it does not violate the above constraint.

Let H be a set of ground atoms, we indicate as $\mathcal{P}(H)$ the set of predicate symbols mentioned in H. For example if $H = \{p(a), q(b), r(c)\}$ then $\mathcal{P}(H) = \{p, q, r\}$. If H is consistent, it will be called an *assignment* to $\mathcal{P}(H)$.

Definition 2.2 *Given a diagnostic problem $DP = \langle BM, HYP, CXT, \langle \Psi^+, \Psi^- \rangle\rangle$, an assignment $H \subseteq HYP$ such that $\mathcal{P}(H) = \mathcal{P}(HYP)$ is a diagnosis for DP if and only if*

$$\forall m(x) \in \Psi^+ \; BM \cup CXT \cup H \vdash m(x)$$

$$\forall m(y) \in \Psi^- \; BM \cup CXT \cup H \nvdash m(y)$$

3 Complexity Results

In [12] it is shown that, apart from particular restrictions, solving an abductive problem is in general an NP-hard problem. Diagnostic problems satisfying definition 2.1 can be viewed as a kind of problems classified in [12] as *incompatibility abduction problems*: incompatibility relations are represented by the fact that different ground instances of the same abducible predicate are incompatible. In the following, when we will refer to a diagnostic problem we will consider a problem satisfying definition 2.1.

Definition 3.1 *Given a diagnostic problem $DP = \langle BM, HYP, CXT, \langle \Psi^+, \Psi^- \rangle\rangle$ and an assignment to abducibles $H \subseteq HYP$ with $\mathcal{P}(H) = \mathcal{P}(HYP)$:*

- *CONCHECK is the problem of deciding whether H is consistent with the observations (i.e. whether $\forall m(x) \in \Psi^- \; BM \cup CXT \cup H \nvdash m(x)$);*
- *COVCHECK is the problem of deciding whether H covers the observations to be accounted for (i.e. whether $\forall m(x) \in \Psi^+ BM \cup CXT \cup H \vdash m(x)$);*
- *DIAGCHECK is problem of deciding whether H is a diagnosis to DP (i.e. DIAGCHECK=CONCHECK+COVCHECK).*

Since it is well-known that verifying whether a given atom is a consequence of a set of propositional definite clauses is linear in the size of the set of clauses [16] (and since our system model BM is equivalent to a propositional definite clause theory), we have the following proposition.

Proposition 3.1 *CONCHECK, COVCHECK and DIAGCHECK are in P*

Let us then consider the following decision problem.

Definition 3.2 *DIAGSAT is the decision problem consisting of determining whether an instance of a diagnostic problem $DP = \langle BM, HYP, CXT, \langle \Psi^+, \Psi^- \rangle\rangle$ has a solution.*

In [12], incompatibility abduction problems have been proved to be NP-hard in general, with some special class of incompatibility problems (i.e. *independent incompatibility abduction problems*) being NP-complete. We can show that the NP-completeness property also holds for DIAGSAT.

Theorem 3.1 *DIAGSAT is NP-complete.*

(see the appendix for the proof)

Let us then define what we mean by diagnosis adaptation problem. We are essentially interested in studying adaptation strategies that can be classified as replacement of abducibles (interpreted as deletion of an abducible followed by the addition of a new one). As also noticed in [31], a case-based system adopting a conservative approach tries to re-use as much as possible of the retrieved solution to be adapted; we will then consider the following problem.

Definition 3.3 Diagnosis Adaptation Problem. *DASAT is the decision problem defined as follows: given a diagnostic problem* $DP_1 = \langle BM, HYP, CXT_1, \langle \Psi_1^+, \Psi_1^- \rangle \rangle$*, a diagnosis* H *to the problem* $DP = \langle BM, HYP, CXT, \langle \Psi^+, \Psi^- \rangle \rangle$ *and an integer* $k \leq |H|$*, determine whether there exists a diagnosis* H' *to* DP_1 *containing a sub-assignment of* H *of cardinality at least* k*.*

In order to transform H into H', we have to define an *adaptation strategy* $\mathcal{A}$. Such a strategy has to determine which abducibles to be deleted and which others to be added to H, in order to obtain H'; the number of abducibles that $\mathcal{A}$ replaces must be at most $|H| - k$ abducibles (in order to have H' containing a sub-assignment of H of cardinality at least k).

Theorem 3.2 *DASAT is NP-complete.*

(see the appendix for the proof)

If we consider DA1SAT to be the decision problem DASAT restricted to the case where the set $\Psi_1^+ = \Psi^+ \cup \{\alpha\}$ (i.e. observations to be accounted for in the new problem differ only for one atom from those of the old problem), then we obtain the following corollary.

Corollary 3.1 *DA1SAT is NP-complete*

Theorem 3.2 and Corollary 3.1 are particularly relevant, since they show that the problem of adapting in a conservative way (i.e. by re-using as much as possible of the retrieved solution) can be as hard as generating the new solution from scratch, even if the retrieved and the current case are very similar. This result may seem to clash with the intuition that the re-use of solution for a similar problem is always easy. However, this result is not peculiar to diagnostic problem solving: Nebel and Koehler have carried an in-depth analysis of case-based planning and they have shown that the theoretical computational complexity of conservatively adapting plans of previously solved problems is in general at least as hard as generating the plans from scratch [31]. In particular, results analogous to Theorem 3.2 and Corollary 3.1 in [31] are still worse, since they show that for a planning problem for which plan generation is polynomial, plan adaptation is NP-complete.[3]

The above results are important since they show that there is no guarantee that a CBR approach is in general less expensive from a computational point of view than MBR, and therefore any multi-modal reasoning architecture for diagnosis based on the re-use of past solution has to be carefully conceived. The potential for a significant improvement in terms of speed-up for complex tasks like diagnosis and planning by integrating CBR with problem solving from first principles is supported by empirical studies (particular relevant for this point are the results obtained by Veloso in planning [50] and by Koton in diagnosis [24]). In the rest of the paper we discuss under which conditions we have good chances that a multi-modal architecture combining CBR and MBR provides better performance that a pure MBR system in diagnostic problem solving. This analysis is carried on by taking into consideration ADAPtER, an architecture integrating a case-based module with a model-based diagnostic module, firstly presented in [32].

[3] In citeAu:02, the authors show that the results of Nebel and Koeler do not apply to the (non conservative) plan adaptation via Derivational Analogy. See Section 8 for a brief comparison with this work.

4 The ADAPtER System

The name **ADAPtER** is an acronym for **A**bductive **D**iagnosis through **A**daptation of **P**ast **E**pisodes for **R**e-use and indicates a diagnostic architecture combining model-based reasoning and case-based reasoning in a uniform and flexible framework based on the well-founded specification of the notion of diagnosis specified in Section 2. A peculiar feature of the system is the use of two different knowledge bases:

- a CASE MEMORY containing a set of solved diagnostic problems with the corresponding solutions; each stored case is characterized as $C = \langle CXT, \langle \Psi^+, \Psi^- \rangle, SOL \rangle$ where CXT is the contextual information under which the case has been solved, Ψ^+ is the set of manifestations covered by the solutions of the case, Ψ^- is the set of manifestation that the case solutions must not imply and $SOL = \{H_1, \ldots H_n\}$ is the set of possible diagnoses (i.e. solutions) of the case;
- a BEHAVIORAL MODEL corresponding to the domain model used by the model-based component for solving a problem from scratch.

From the architectural point of view, **ADAPtER** involves the set of components shown in figure 1 where links represent data flow. The high-level behavior of ADAPtER can be described

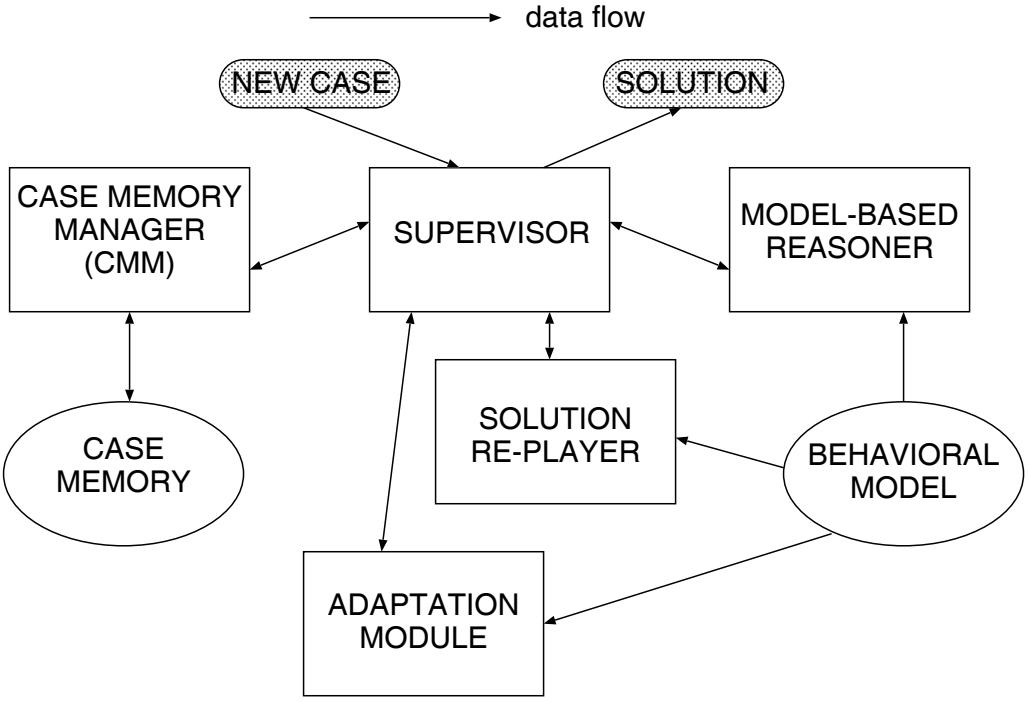

Figure 1: ADAPtER Architecture

by the pseudo-code in which we assume that for each module `M`,

```
M(input:...;output:  m-solutions,...)= true iff m-solutions is not empty
```

Important parameters of the modules of the architecture are a set of thresholds `S, T1, T2, T3, T4`. Their meaning will be clarified while describing the architecture; for now it is worth

noting that parameters `T1` through `T4` are temporal thresholds representing specific time-outs on the execution of the relative modules, while `S` is peculiar to the retrieval step and concerns the adaptability of a retrieved case with respect to the input one. It is worth noting that the set of `m-solution` is considered empty also if a corresponding time-out has been reached. ADAPtER starts with a time limit `T1` (i.e. no more that `T1` time instants are allowed for the solving a problem); each specific module receiving a time limit `Ti` as input will return a new time limit `Tj` to be used by the next module, representing the remaining available time.

```
ADAPtER(new-case,Case-Memory,Behavioral-Model,S, T1):
IF NOT RETRIEVE(input: new-case, Case-Memory, S, T1
                output: retrieve-solutions, T2)
   THEN IF MBR( input: new-case, Behavioral-Model, T2
                output: mbr-solutions);
        THEN return(mbr-solutions)
        ELSE return("failure")
   ELSE IF OK-SOLUTION(input: new-case, retrieve-solutions, Behavioral-Model, T2
                  output: replayed-solutions, T3)
     THEN return (replayed-solutions)
     ELSE
      IF ADAPTATION(input: new-case, replayed-solutions, Behavioral-Model, T3
                   output: adaptation-solutions, T4)
         THEN return (adaptation-solutions)
         ELSE
          IF  MBR(input: new-case, Behavioral-Model, T4
                 output: mbr-solutions);
          THEN return(mbr-solutions)
          ELSE return("failure")
```

When presented with a new case $DP = \langle CXT_1, \langle \Psi_1^+, \Psi_1^- \rangle \rangle$, the SUPERVISOR first invokes the CASE MEMORY MANAGER (CMM) in order to retrieve the most promising cases from the CASE MEMORY (**RETRIEVE**). Such a step evaluates the degree of match between DP (i.e. the current case to be solved) and the cases in the memory, using a heuristic function which estimates the adaptation effort rather than just the similarity between the current and the retrieved cases. **RETRIEVE** returns the solutions of cases in the case memory with the lowest value of such heuristic function or returns a failure if it is unable to find in the case memory a case solution sufficiently easy to be adapted to the input case DP (i.e. there is no case in the case memory suitable to be adapted to the input case, since the heuristic function reports a value greater than the threshold `S` for any case in the case memory).

In particular, a retrieval algorithm called *Pivoting-Based Retrieval* (PBR) [36] has been developed for implementing **RETRIEVE**. Since the description of PBR is out of the scope of the present paper, it is just worth noting that PBR is a form of Adaptation-Guided Retrieval [42] and exploits a heuristic function able to estimate the cost of adaptation; this is possible, since such function has sufficient knowledge for determining what kind of reasoning mechanism the adaptation module has to invoke, in order to solve the discrepancies between the observations of the input case and the ones of the case stored in the case memory. The PBR algorithm is based on the computation of suitable bounds on the adaptation cost of each solution of a case; such bounds are then used to restrict the search for the best case solution to be retrieved. We have shown that the search strategy of PBR is admissible, i.e. we are guaranteed to find out the case solutions which minimizes the heuristic function (see [36] for more details).

If **RETRIEVE** fails, the control is switched directly to the MODEL-BASED REASONER (**MBR**). On the contrary, in case **RETRIEVE** succeeds, this module returns a set $\{H_1 \dots H_m\}$ representing the best solutions retrieved from the case memory (i.e. those having minimal estimated adaptation effort, with respect to the input problem DP).

For each $H_i \in \{H_1 \dots H_m\}$ the SOLUTION RE-PLAYER is invoked by the SUPERVISOR to replay the retrieved solution H_i (**OK-SOLUTION**). Let $DP_1 = \langle BM, HYP, CXT_1, \langle \Psi_1^+, \Psi_1^- \rangle \rangle$ be the input problem; the retrieved solution H_i is used together with the contextual data CXT_1 of the case under examination DP_1 and the BEHAVIORAL MODEL BM (which at the logical level is modeled as a set of definite clauses) to recompute all the possible consequences[4]. This step computes the transitive closure, in terms of predicate symbols, of the set of atoms $CXT_1 \cup H_i$, by using the model theory BM; moreover, for each predicate $p(a)$ such that $BM \cup CXT_1 \cup H_i \vdash p(a)$, it stores the *support* $\mathcal{S}(p(a))$, i.e. the set of abducibles in H_i that are necessary in at least one derivation of $p(a)$ from the theory $BM \cup CXT_1 \cup H_i$ (as illustrated in the following, this information is used by the adaptation mechanism). Formally, if $BM \cup CXT_1 \cup H_i \vdash p(a)$, the support for $p(a)$ is defined as:

- if $p(a) \in H_i$, $\mathcal{S}(p(a)) = \{p(a)\}$;
- if $p(a) \notin H_i$, let $\alpha_1 \rightarrow p(a), \dots, \alpha_n \rightarrow p(a)$ be the clauses in BM whose head is $p(a)$ and such that $BM \cup CXT_1 \cup H_i \vdash \alpha_j$ $(j = 1, \dots, n)$ and $p_1(a_1), \dots, p_k(a_k)$ be the (non context) atoms occurring in the bodies α_j; the support for $p(a)$ is $\mathcal{S}(p(a)) = \bigcup_{l=1}^{k} \mathcal{S}(p_l(a_l))$.

The **OK-SOLUTION** step succeeds if *consistency* and *covering* between the solution's predicted observable parameters and the current set of observations are satisfied. More technically we have to verify whether

$$\forall m(y) \in \Psi_1^- \quad BM \cup CXT_1 \cup H_i \nvdash m(y) \tag{1}$$

$$\forall m(x) \in \Psi_1^+ \quad BM \cup CXT_1 \cup H_i \vdash m(x) \tag{2}$$

It is worth noting that **OK-SOLUTION** is the implementation of the DIAGCHECK problem described in Section 3 that we know to be efficiently solvable.

A failure occurs in **OK-SOLUTION** if and only if every retrieved solution H_i does not satisfy condition 1 and 2; in such a case, one of the replayed solution is passed on to the ADAPTATION MODULE for the **ADAPTATION** step. Actually, ADAPtER can choose whether to try to adapt one randomly chosen best solution or to try to adapt more than one retrieved solution[5]. In the following we will consider the first case for the sake of simplicity; this step tries to adapt the retrieved solution to be a solution of the current case, by using the same domain knowledge (that is the BEHAVIORAL MODEL) used by the MODEL-BASED REASONER.

[4] CXT_1 is potentially different from the contextual data of the diagnostic problem which H_i refers to.

[5] **RETRIEVE** always returns all the solutions with the minimum estimated adaptation effort (provided this is less than the threshold S), therefore all the retrieved solutions $\{H_1, \dots, H_m\}$ are equally suitable to be adapted. Three different policies have been defined: (1) only one retrieved solution is selected for adaptation; (2) the adaptation step selects a retrieved solution at a turn and it stops when either the adaptation succeeds for one of them or when all the retrieved solutions have been considered; (3) the adaptation is always attempted for all the retrieved solutions. This last policy could improve the *quality* (defined in Section 5.1) of the set of computed solutions.

As before, let $DP_1 = \langle BM, HYP, CXT_1, \langle \Psi_1^+, \Psi_1^- \rangle \rangle$ be the input problem. Given the replay of a retrieved solution H_i in the new context CXT_1 computed by **OK-SOLUTION**, in order to describe the adaptation mechanism, we define the following sets:

- $O_E = \{m(a)/\ m \textit{ is an observable parameter and } BM \cup CXT_1 \cup H_i \vdash m(a)\}$ is the set of the values of observable parameters (manifestations) entailed by the retrieved solution H_i in the new context CXT_1 (i.e. the set of manifestations occurring in the transitive closure computed by **OK-SOLUTION**);
- $O_{CONFLICT} = \{m(a)/\ m(a) \in O_E \textit{ and } m(a) \in \Psi_1^-\}$ is the set of manifestations occurring in the transitive closure and conflicting with the observations for the input case;
- $O_{NEW} = \{m(a)/\ m(a) \in \Psi_1^+ \textit{ and } m(a) \notin O_E\}$ is the set of manifestations that are not present in the transitive closure, but that should be covered in the current problem;
- $A_S = \bigcup_{m(a) \in O_E} \mathcal{S}(m(a))$ is the set of abducibles in H_i supporting the manifestations entailed by the retrieved solution in the new context CXT_1[6];
- $\overline{A}_S$ is the complement of A_S w.r.t. H_i.

The goal of the adaptation is to remove possible inconsistencies in the replayed solution and to build the missing explanations (coverings) for some manifestations of the input case, in order to obtain a formally correct solution.

The adaptation process works in two steps (for the details see [37, 32]):

1. **Inconsistency removal.** If $O_{CONFLICT} \neq \emptyset$, than the replayed solution does not satisfy condition (1) and consistency must be re-established. This mechanism disproves the explanation leading to each $m(a) \in O_{CONFLICT}$, by removing from H_i a set of abducibles responsible for such an inconsistency. For each $m(a) \in O_{CONFLICT}$, the abducibles to be removed are searched in those ones occurring in the support $\mathcal{S}(m(a))$ of $m(a)$.
2. **Explanation Construction.** Let R be the set of abducibles removed from H_i by the previous step. The Explanation Construction step keeps fixed the set $A_S - R$ of abducibles and searches for an assignment K for the predicates in $\mathcal{P}(\overline{A}_S \cup R)$ such that $(A_S - R) \cup K$ is a solution to the input case. To perform its task, this step uses the same procedure used by the MODEL-BASED REASONER. In particular, in order condition (2) to hold, this mechanism builds abductive explanations for all observations in O_{NEW} and for those ones in Ψ_1^+ that are no longer supported by abducibles because of the inconsistency removal step (i.e. such that $A_S - R$ is not an abductive explanation for them).

It is worth noting that there is no guarantee that **ADAPTATION** actually generates all the minimal solutions (i.e. solutions containing the minimal number of abducibles representing faults) potentially provided by **MBR**. We will investigate this problem more deeply in Section 5, however we have to notice here that **ADAPTATION** does not necessarily provide a single solution, because there are many ways for constructing explanations. Among the alternatives that are considered, only the minimal ones are actually produced by **ADAPTATION**. Moreover,

[6] It is worth noting that there can be some abducibles in H_i that do not belong to the support of any manifestation in O_E. For instance, if BM is a fault model, the abducibles in H_i representing normality do not participate in any derivation of the manifestations in O_E, therefore they do not occur in any support.

the adaptation mechanism does not explores all the different ways of removing inconsistencies. Instead, it makes use of some heuristics to choose the set R of abducibles to be removed [37, 32]. It may happen that the set $A_S - R$ of the abducibles retained in the adaptation step prevents **ADAPTATION** to find a solution; in this case, **ADAPtER** can either try to adapt another retrieved solution or it can switch to **MBR** to solve the input case. As we said, in the present work we consider the latter policy for our analysis. Therefore, if the attempt to re-use past solved problems has failed (i.e. either **RETRIEVE** or both **OK-SOLUTION** and **ADAPTATION** have failed), **MBR** is invoked and **ADAPtER** tries to solve the diagnostic problem from scratch with a given time limit (see Section 4). **MBR** implements the characterization of diagnosis introduced in Section 2 and it is able to provide all minimal diagnoses which cover the manifestations to be explained, consistently with all the observations. In case the diagnostic problem is not solved within the time limit, **MBR** returns a failure, even if the problem could be solved with more resources.

The description of the architecture reported above should have made clear an important characteristic of ADAPtER: both the CBR and the MBR components share the same domain knowledge, in particular the same behavioral model. In this way, there is no need of eliciting from domain experts additional knowledge for performing adaptation. More important, the same notion of solution applies both to the results provided by CBR and the ones provided by MBR. For some aspects, we could consider the **OK-SOLUTION** and **ADAPTATION** modules as specialized mechanisms for performing focused MBR.

The described problem solving architecture is complemented (at a meta level) by a learning architecture working off-line with respect to the problem solving cycle. Indeed, in case **MBR** provides a solution, the case that has been solved is learnt together with its solutions in the case memory. Different learning strategies have been tested in ADAPtER, taking explicitly into account the utility problem and revising the content of the case memory depending on the case that is currently learnt. In particular, cases are not only added to case memory, but specific case deletion or case replacement strategies can be used. Results concerning the efficacy of such strategies are deeply discussed and described in [35]. Figure 2 shows the details of the CASE MEMORY MANAGER by considering its internal structure: the RETRIEVER aimed at implementing the **RETRIEVE** step (retrieving the set of cases $(C_1, \ldots C_n)$ with associated solutions from the case memory in response to the **SEARCH(DP)** command aimed at searching the case memory for the n most adaptable case solution with respect to the current diagnostic problem DP) and the LEARNER whose main task is to interact with the SUPERVISOR, in order to receive information on which case to add to the case memory and which other cases to remove from it, as a consequence of a learning operation. In particular, when the SUPERVISOR send a **LEARN(C)** command to the learner, the new case C is added to the case memory and a set of cases $(C'_1 \ldots C'_k)$ are possibly removed from the case memory, depending on the case maintenance strategy adopted (see [35]). The SUPERVISOR invokes the LEARNER only after a case has been solved by the MODEL-BASED REASONER; in terms of classification of case memory management strategies this corresponds to an *off-line integration type approach* [51].

In summary, by considering the multi-modal architecture introduced, we can notice that there are two main issues that are worth to be considered:

- the **MBR** process may be computationally intractable in some cases, so the possibility of solving a new problem in a different, possibly integrated way can be worth to be pursued; this is true because if a good case memory is available, the need for adaptation is lowered

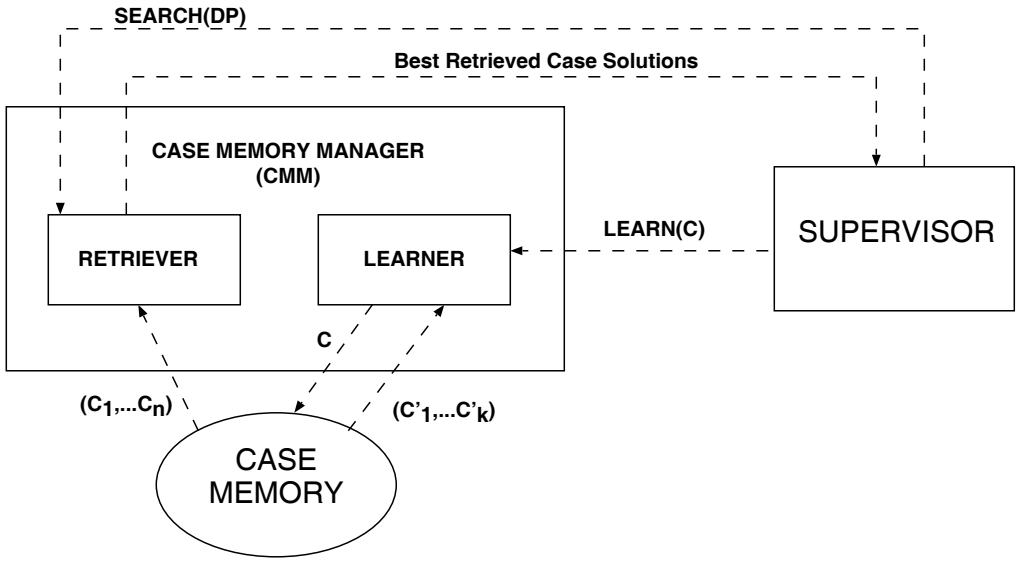

Figure 2: Case Memory Manager Architecture

and a simple (and computationally tractable) solution check can be performed. Even in the situation where adaptation is needed, the worst case complexity of the adaptation process seems not to be the usual case [33, 34, 35];

- in order to be able to deal with different kinds of cases (even inside the same domain), the system is not provided with any off-line *training phase* where the case memory is built; on the contrary the system is able to start with an empty memory and, through a continuous learning process, to adapt itself to every change in the characteristics of cases to be solved[7];

Next sections will address different ways of multi-modal integration, in order to take into account such issues.

5 Experimental analysis of ADAPtER

In this section we aim at showing to what extent a multi-modal architecture like ADAPtER can achieve better performance with respect to a pure MBR architecture. However, the analysis is complex since the term "performance" is very general and different specific issues related to this aspect can be identified.

5.1 The different facets of performance

Although computation time is of paramount importance in characterizing the performance of a system, other parameters have to be taken into account: it is possible that different architectures may exhibit different levels of *competence* in the problem space, so performance is measured as the percentage of problems that can be solved with respect to the whole problem space;

[7] In diagnostic problem solving, for instance, the number of faults in the solution of a given problem may vary significantly during the lifetime of the system to be diagnosed; a case-based diagnostic architecture with a considerable adaptativity can be successfully adopted in this situation.

moreover, different problem solving strategies may provide solutions of different *quality*, because for instance, different approximations or heuristics are used, so in this case performance is measured as a suitable metric with respect to optimal solutions. In general it is believed that approaches based on first-principle guarantee optimal competence (all the problems that can be solved by using the domain knowledge are actually solved) and quality (the first-principle problem solver gets optimal solution), while they may result expensive from a computational point of view. We have then to characterize precisely the notion of computation time, quality and competence.

First of all, let us examine competence. The CBR module has a reduced competence with respect to MBR: while the MBR component is in principle able to solve any diagnostic problem in the modeled domain (as far as the problem description is not inconsistent with respect to the domain theory); this is not true for the CBR component whose competence strictly depends on the content of the case base as well as from the available adaptation knowledge.

However, in practice the competence of MBR is not optimal since diagnostic problem solving has to be performed under limited resources and computation time is a critical resource (especially when considering diagnosis of real-time systems). In case the diagnostic system is asked to provide a solution within a predefined time constraint, the *practical competence* may be different from *theoretical competence*. In such situations, the practical competence of the MBR module may be quite far from the theoretical one, as there may be diagnostic problems that are not solvable within the specified time limit. In our architecture we have introduced some time thresholds representing the maximum effort in computation time to be spent for solving a diagnostic problem. If the threshold is reached without producing the diagnostic solutions, the problem is considered unsolvable. In this way, in carrying out an experimental analysis we can consider that both ADAPtER and its pure MBR module are incomplete.

With respect to quality, while a model-based approach is able to provide all minimal diagnoses for a specific diagnostic problem, we have already pointed out that this is not guaranteed by ADAPtER (see Section 4). However, part of the problem is intrinsic to the CBR approach, since the input problem is solved by first retrieving and then adapting a solution of another problem. If the input problem has a large number of possible alternative solutions, it is clear that just adapting a single solution of a similar problem has limited chances of getting all the solutions to the input problem[8]. For this reason the set of diagnoses obtained by the CBR component on a given problem may not cover all the possible solutions.

To actually measure the impact of the multi-modal architecture on the quality parameter, we have first to select a gold standard and then to compare how well the solutions provided by a problem solver PS' fit the solutions provided by the problem solver PS selected as the gold standard. Since MBR produces an optimal set of solutions for each diagnostic problem that it is able to solve (i.e. whenever a diagnostic problem is solved by MBR, *all* the *minimal diagnoses* are returned), MBR has been selected as reference point. Therefore, we operalize the notion of quality by comparing the set of solutions $SOL_{PS'}(DP)$ provided by the PS' problem solver for any diagnostic problem DP with $SOL_{MBR}(DP)$ produced by MBR for the same diagnostic problem whenever DP is solved both by PS' and by MBR [9].

[8]Some CBR systems may address this problem by combining multiple cases (and multiple solutions) to solve the input problem, however if the given task or domain is not modular, very complex adaptation strategies will be needed to get all possible solutions.

[9]The cases in which either PS' or MBR failed in solving the diagnostic problem are taken into consideration by the *competence* parameter.

If we instantiate the general notion of quality in case of ADAPtER, the quality $Q_{ADAPtER}(DP)$ for the diagnostic problem DP is computed in the following way:

$$Q_{ADAPtER}(DP) = \begin{cases} \frac{|SOL_{ADAPtER}(DP) \cap SOL_{MBR}(DP)|}{|SOL_{MBR}(DP)|} & \text{if } SOL_{ADAPtER}(DP) \neq \emptyset \\ & \text{and } SOL_{MBR}(DP) \neq \emptyset \\ undefined & otherwise \end{cases}$$

where $SOL_{ADAPtER}(DP)$ and $SOL_{MBR}(DP)$ are the set of solutions for the diagnostic problem DP determined by ADAPtER and MBR respectively.

It is worth noting that this way of defining the *quality* parameter is rather penalizing for the multi-modal architecture, since it ignores any non-minimal solution possibly provided by it. A more realistic evaluation of the quality could be obtained in restricting the comparison to cases where MBR provides just a single solution. In [35] we have analyzed such a case and the experimental results show that the quality of the solutions of ADAPtER is rather satisfying.

5.2 Characterizing the experimental setting

Since theoretical results reported in Section 3 prevent us from claiming that reuse of diagnostic solutions is, in general, simpler than solving diagnostic problem from scratch, we have carried on a set of experiments, in order to evaluate and compare the performance of ADAPtER with respect to the pure MBR component. Moreover, we have seen that performance cannot be reduced to a single parameter. For this reason the experiments should be able to take into account the different aspects of performance and possibly to provide material for explaining some of the relations among them.

In defining the experimental setting, we have to consider:

- the domain(s) used as test-bed
- the test sets used for the experimental analysis
- the setting of the controlled parameters which can influence the performance of the diagnostic system

We have performed experiments in two different domains: $D1$ represents a significant portion of a fault model of an industrial plant, whereas $D2$ concerns a (relatively simple) domain of car faults. $D1$ is more complex than $D2$ since the domain model of $D1$ involves 31 components with a total number of 100 possible faults, while in $D2$ we have ten components for 39 possible faults. Moreover, in $D1$ we have 46 observable parameters with 104 different manifestations, whereas in $D2$ the observable parameters are 14 and the different manifestations 37.

In both domains the knowledge available concerns just the faulty behavior. In other words, the domain model contains definite clauses relating the presence of faults in the different components or parts of the system to be diagnosed with their consequences (both direct and indirect). As shown in [14] and [27] the type of domain knowledge available has to play a major role in selecting the appropriate notion of diagnosis. Since we had at disposal fault models of the domains, the set of observations OBS relevant to a diagnostic problem can be partitioned into the two sets OBS^A and OBS^N of the abnormality and normality observations, respectively (with $OBS^A \neq \emptyset$). Furthermore in our experiments we chose $\Psi^+ = OBS^A$ since we are interested in diagnoses that covers all the abnormality observations consistently with all the observations (see definition 2.2).

	$IS1$	$IS2$	$IS3$	$IS4$	CE
Domain	$D1$	$D1$	$D1$	$D1$	$D2$
Cardinality of the set	2000	2000	2500	3000	2000
Max no. of injected faults	4	2	3	3	4
P_{NORMAL}	0.8	0.8	1	0.8	0.8

Table 1: Parameters characterizing the test sets

5.3 The Test Sets

In our analysis we had to face the lack of representative sets of diagnostic problems solved by human experts (in particular, in the industrial domain, we did not have access to them). Even when a set of solved cases is available, its use for evaluating the performance of a system could not be straightforward, since the characteristics of the problems (i.e. the value of any parameter that can have an impact on the performance) could be unknown.

Both the lack of real world solved cases and the need of a precise knowledge and control on the cases of the test set have suggested the automatic synthesis of test sets by means of a simulator of the system to be diagnosed in presence of injected faults. In particular, the simulator is able to generate a case description by predicting, using the specific fault model of the domain we are considering (either industrial plant or car faults), the values of the observable parameters when a specific set of faults is assumed and the contextual situation is specified.

The characteristics of a batch of cases automatically generated by the simulator can be controlled by means of suitable parameters; the most significant one we have used in the experiments reported in the present paper is the maximum number of injected faults (that is, the maximum number of faults to be included in the case). This parameter is very relevant because the larger is the number of faults, more complex is the case description and harder is the task of the diagnostic system (this phenomenon is well known and one of the reasons for adopting MBR techniques in diagnostic problem solving is the ability of dealing with multiple faults).

Another aspect which can influence the number of alternative diagnoses produced by a diagnostic problem solving is the number of observations available for the case description. In particular when the case description is partially incomplete (that is, only some of the observable parameters have an observed value in the case under examination, while for the other manifestations the value is unknown), the number of alternative diagnoses is very large and the diagnostic problem may not be considered totally solved, since further observations may reduce the set of diagnoses and in some cases radically change them. For this reason we have varied the degree of completeness of the case description (in terms of observations) in the test sets.

Table 1 reports the main characteristics of the test sets and of the parameters used to generate them. We have generated 5 test sets: 4 for domain $D1$ and just one for domain $D2$. Each test set is quite large and involves cases for which at least one minimal solution exists since each diagnostic problem has been generated via the injection of at least one fault. Test sets have different distributions of cases since we have varied the maximum number of faults injectable in each case. Test sets vary also for the way the set of observations characterizing the cases are determined. Given the injected faults and the set of contextual information CXT, the simulator determine the transitive closure using domain theory DT and therefore provides

CPU time	$IS1$	$IS2$	CE
[0 msec.,200 msec.)	36.95%	54%	63.65%
[200 msec.,500 msec.)	13.55%	18.1%	2.75%
[500 msec.,1 sec.)	8.6%	9.55%	18.5%
[1 sec.,10 sec.)	20.8%	13.55%	11.2%
[10 sec.,1 min.)	7.2%	2.8%	3.0%
[1 min.,10 min.)	6.0%	1.0%	0.85%
[10 min.,1 h.]	2.6%	0.5%	0%
timeout	3.1%	0.35%	0.0%
out of memory	1.2%	0.15%	0.05%

Table 2: DP distribution according MBR_time(DP)

the set of abnormal manifestations produced by those faults. All these abnormal manifestations OBS^A are included in the case description. Since we have at disposal only fault model the simulator cannot directly determine OBS^N; it is build by inserting into it (with probability P_{NORMAL}) a manifestation $m(normal)$ any time there is no prediction made by simulator for manifestation m. In all test sets, but $IS3$, for each diagnostic problem, the 80% (in the average) of these manifestations are assumed to occur as normal, whilst in $IS3$ *all* the manifestations have a known value i.e., P_{NORMAL} is set equal to one (see Table 1).

We have generated different sets for domain $D1$ since we are interested in understanding the influence of different characteristics on the performance of the diagnostic system. More important, we are interested in a fair evaluation of the performance of a multi-modal reasoning system with respect to a pure MBR system; for this reason we have decided to use $IS1$ and $IS2$ as training sets to infer the values of the parameters governing the control strategy of ADAPtER. Once the parameter values have been determined, we have run $IS3$ and $IS4$ for the actual evaluation of ADAPtER. It is worth noting that $IS3$ and $IS4$ have different characteristics from $IS1$ and $IS2$, in particular they differ for the max number of injected faults in each diagnostic case.

5.4 Experimental results

The first experiment we have performed concerns the actual difficulty of solving diagnostic cases by a pure MBR system. Since theory predicts that we would have to deal with hard problems, we are interested in measuring whether intractability occurs and how frequent it is. For this reason we have run the MBR problem solver on the test sets $IS1$, $IS2$ and CE by putting for each problem in these sets a time-out of 3.600.000 msec. of CPU time (i.e. 1 hour) on a PENTIUM II with 128 MB RAM under Solaris operating system[10]. For each solved problem DP, we collected the time $MBR\_time(DP)$ spent to solve it. Table 2 reports how the problems of each set are distributed according to $MBR\_time(DP)$. For example, we can see that 6% of the problems in $IS1$ require a MBR time included in the interval [1 min.,10 min.) (i.e. the MBR problem solver has consumed a CPU time between 1 min. and 10 min. for solving each of them), whereas 54% of the problems in $IS2$ take less than 200 msec. to be solved. The results reported in Table 2 show that there are problems that are very hard to solve (the combinatorial explosion predicted

[10]All the different diagnostic problem solvers referred to in this paper had been implemented using SICStus Prolog.

by theoretical results actually occurs). It is worth noting that the number of hard problems is larger in $IS1$ (where the maximum number of injected faults is 4), but unfortunately it is not negligible in $IS2$, despite each $DP \in IS2$ has been generated by injecting a maximum of two faults. As expected the problems in CE are easier to be solved because domain $D2$ is simpler; however also in CE there are (few) problems that require a huge amount of computational resources. It is worth noting that also space shortage could prevent some problem to be solved (see the row 'out of memory' in Table 2).

We have also run ADAPtER on $IS1$, $IS2$ and CE in order to estimate some important parameters which actually influence the behavior of the diagnostic system. In particular, the threshold S is used in the **RETRIEVE** step for deciding whether the retrieved case with the best estimate of the adaptation effort is sufficiently close to the input case C to be used for adaptation. As a result of the experiments on $IS1$, $IS2$ and CE we have determined a value for S corresponding to a medium adaptability effort for the considered domain[11]. Actually, the experiments have allowed to estimate parameters governing the learning component of ADAPtER (in particular, the case deletion mechanism). The discussion on such parameters is outside the scope of the paper and details can be found in [35].

A critical parameter for evaluating the performance of ADAPtER in comparison with the MBR concerns the threshold `T1` i.e. the maximum CPU time allocated for solving the diagnostic problem under consideration.

We have run a set of experiments on test sets $IS3$, $IS4$ and CE for evaluating speed-up, competence and quality of ADAPtER. In all these experiments, we have set `T1` to 60 seconds of CPU time on a PC Pentium II. Therefore, if a diagnostic problem C belonging to the test set $IS3$, $IS4$ or CE has consumed a CPU time larger that 60 seconds and ADAPtER has not provided a solution for C, the problem is considered unsolved by ADAPtER. The same criterion is applied for MBR. It is apparent that the lower is `T1`, the lower is the competence level, however it will be shown that the competence of ADAPtER and MBR are significantly different by using the same threshold.

Table 3 reports the results on the average computation time needed for solving (or trying to solve) the problems in the test sets $IS3$, $IS4$ and CE. It is worth noting that as regards computation time ADAPtER outperforms a pure MBR approach. This is not only true for test sets in domain $D1$, but also for test set CE whose problems are taken from the much simpler domain $D2$.

A very positive result for ADAPtER concerns its competence. Despite the fact that MBR should have a perfect competence (if infinite resources are available), it turns out that ADAPtER is able to solve much more cases than MBR is. The experiments show that several times in domain $D1$ MBR takes too long time for searching for a solution (i.e. the time out of 60 sec. is reached without finding a solution). On the contrary ADAPtER exploits cases already solved for solving most of the cases submitted to it. It is also worth noting that these results are obtained starting from an empty case memory and the learning module of ADAPtER is able to learn suitable cases so that the competence is very high.

The value for the timeout threshold (here set to 60 seconds) strongly depends on the time requirements the system must fulfil. Figure 3 depicts the competence of the two architectures corresponding to different timeout thresholds less than 60 seconds. The figure shows that ADAPtER always outperforms MBR. In particular, the advantage of ADAPtER over MBR ap-

[11] More precisely $S = 30$ for domain $D1$ and $S = 12$ for domain $D2$.

	$IS3$	$IS4$	CE
MBR	4269.7 $\pm$516.8	6468.8 $\pm$580.6	1789.3 $\pm$304.1
ADAPtER	1525.8 $\pm$321.7	1258.7 $\pm$260.6	241.3 $\pm$131.6

Table 3: Comparison between ADAPtER and MBR: 95% confidence intervals for the average CPU time.

	$IS3$	$IS4$	CE
MBR	95.32%	92.90%	99.10%
ADAPtER	98.32%	98.73%	99.80%

Table 4: Comparison between ADAPtER and MBR: percentage of solved problems (competence).

pears greater with small timeout thresholds. For instance, with the timeout threshold set to 500 msecs., ADAPtER is able to solve 92.84% of $IS3$ problems, 91.13% of $IS4$ problems and 98.55% of CE problems. With the same timeout threshold, MBR is able to solve 67.12%, 60.47% and 66.40% of $IS3$, $IS4$ and CE problems, respectively.

Table 5 reports data about the quality of the solutions provided by ADAPtER. It is worth noting that we have adopted the very restrictive definition of Section 5.1. For example, in $IS4$ 57.3% of the cases solved by both ADAPtER and MBR, the solutions provided by ADAPtER are exactly the same of the (minimal) ones provide by MBR. It is worth noting that the main difference does not concerns the minimality, but the fact that MBR provides all minimal diagnoses whereas ADAPtER solves the case by adapting one (or few) solutions. A more realistic evaluation of the quality could be obtained in restricting the comparison to cases where MBR provides just a single solution. In [35] we have analyzed such a situation and the experimental results show that the quality of the solutions of ADAPtER is rather good.

6 Using a Cost Model as Analytical Tool

The set of experiments we have performed is quite large and in the previous paragraph we have reported some of the most relevant results. In particular, we have shown that the integration of CBR and MBR provides very significant advantages both in term of saving of computation cost and in terms of competence (not an obvious result and formerly noticed by Veloso [50]).

	$IS3$	$IS4$	CE
ADAPtER	0.675	0.573	0.426

Table 5: the quality parameter for ADAPtER

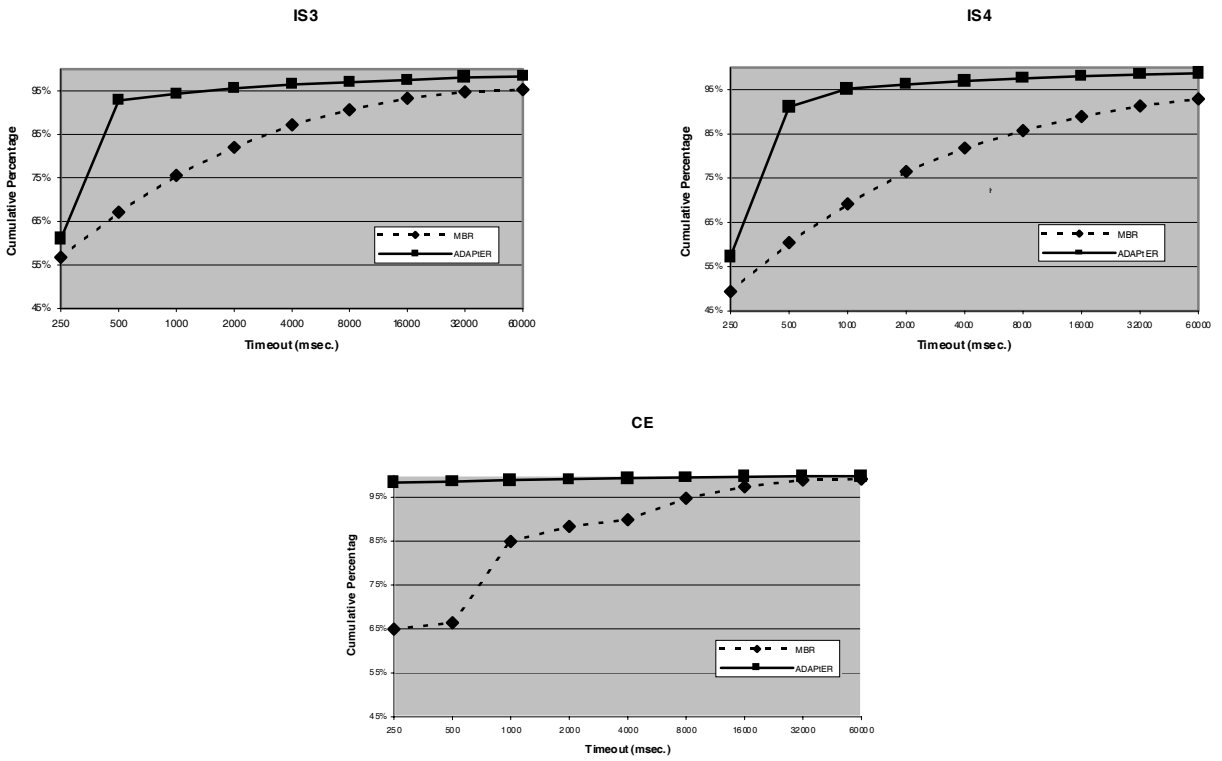

Figure 3: Comparison between ADAPtER and MBR: the competence as a function of the timeout threshold

The set of experiments we have performed has allowed an empirical evaluation of the most critical architectural choices. However, an empirical approach may result very data intensive and extremely expensive from a computational point of view. For this reason we aim at supplementing the empirical approach with an analytical one. A useful approach is based on the development of a cost model, in order to analyze the "global" performance of a system from performance characteristics of its components. The cost model is based on work by van Harmelen [47] and Straatman and Beys [44]. Their model was designed for rule-based and "dynamic" logical languages that include control structures like "sequence" and "while loop". In [48] it has been shown that the cost model can be usefully adopted for evaluating the integration of CBR and MBR components.

A cost model is derived from the control structure of the problem solving architecture and the expected cost of a compound control expression is expressed in terms of its components by considering branches in the control flow. For example, a compound procedure of the form:

```
IF Proc1 THEN Proc2 ELSE Proc3
```

branches after executing `Proc1`. After `Proc1`, either `Proc2` or `Proc3` is executed, depending on the outcome of `Proc1`: success or failure. These occur with a certain probability. This gives the following expression for the expected cost of the compound procedure (`Ps` refers to probability of success and `Pf` to probability of failure):

```
cost(IF Proc1 THEN Proc2 ELSE Proc3) =
cost(Proc1) + (Ps(Proc1)* cost(Proc2)) + (Pf(Proc1) * cost(Proc3))
```

Analogous formulae can be derived for other control structures.

Given the above description it is easy to derive the cost model of ADAPtER by taking into consideration the overall architecture reported in Section 4

```
cost(ADAPtER) =
   cost(RETRIEVE) +
     Pf(RETRIEVE) * cost(MBR_1) +
     Ps(RETRIEVE) * [cost(OK-SOLUTION) +
                       Pf(OK-SOLUTION) * (cost(ADAPTATION) +
                       Pf(ADAPTATION) * cost(MBR_2))]
```

Before addressing the problem how to estimate the parameters appearing in the cost model, it is worth noting that the notion of model can be applied also to "competence" and not only to "computational cost". In particular, we could derive formulas similar to the ones for cost also for competence. For example, given a compound procedure of the form

```
IF Proc1 THEN Proc2 ELSE Proc3
```

the competence can be expressed as

```
competence(IF Proc1 THEN Proc2 ELSE Proc3) =
(Ps(Proc1)* competence(Proc2)) + (Pf(Proc1) * competence(Proc3))
```

where the competence of a basic problem solving method is the probability that the method is able to solve the problem. In case of the ADAPtER architecture, the competence model can be specified as follows

```
competence(ADAPtER) =
     Pf(RETRIEVE) * Ps(MBR_1) +
     Ps(RETRIEVE) * [Ps(OK-SOLUTION) +
                      Pf(OK-SOLUTION) * (Ps(ADAPTATION) +
                                         Pf(ADAPTATION) * Ps(MBR_2))]
```

It is worth noting that the `Ps(proc)` (and `Pf(proc)`) has to be considered in the context where `proc` appears in a program. In the above formula we have denoted with MBR_1 and MBR_2 the invocation of the same component (i.e. MBR) in two different contexts. The first one (MBR_1) refers to the invocation of MBR because of the failure of RETRIEVE and the second one (MBR_2) to the execution of MBR because of the failure of ADAPTATION; we have to consider them as different components as concerns the `cost` (and `Ps`), since the average computational time can be potentially different because the set of cases they operate on are different. For this reason, `Ps(MBR_1)` is actually the probability that MBR succeeds given that RETRIEVE has not been able to retrieve a case. More formally we can interpret

$$\texttt{Ps(MBR\_1)} = Prob(MBR = Success \mid RETRIEVE = fail).$$

In general `Ps(proc)` can be viewed as conditional probabilities with respect to the modules governing the activation of `proc`. Therefore,

Parameter	Estimator
Ps(RETRIEVE)	N_R/N_K
Pf(RETRIEVE)	$(N_K\text{-}N_R)/N_K$
Ps(OK-SOLUTION)	N_C/N_R
Pf(OK-SOLUTION)	$(N_R\text{-}N_C)/N_R$
Ps(ADAPTATION)	$N_A/(N_R\text{-}N_C)$
Pf(ADAPTATION)	$(N_R\text{-}N_C\text{-}N_A)/(N_R\text{-}N_C)$
Ps(MBR_1)	$N_{M1}/(N_K\text{-}N_R)$
Ps(MBR_2)	$N_{M2}/(N_R\text{-}N_C\text{-}N_A)$
Cost(RETRIEVE)	$average(RETRIEVE\_time(case_i))case_i \in S_K$
Cost(MBR_1)	$average(MBR\_time(case_i))\ case_i \in S_K - S_R$
Cost(OK-SOLUTION)	$average(OK - SOLUTION\_time(case_i))\ case_i \in S_R$
Cost(ADAPTATION)	$average(ADAPTATION\_time(case_i))\ case_i \in (S_R\text{-}S_C)$
Cost(MBR_2)	$average(MBR\_time(case_i))\ case_i \in (S_R\text{-}S_C\text{-}S_A)$

Table 6: Estimators of parameters of the cost model

$$\texttt{Pf(ADAPTATION)} = Prob(ADAPTATION = Fail | RETRIEVE = success, OK - SOLUTION = fail).$$

The probabilities and costs of the basic components in the ADAPtER architecture have to be obtained empirically by running the system on a suitable test set.

Let us consider a test set S_K composed of N_K cases and let us assume that S_R is the set of N_R cases from S_K that were successfully retrieved. Let us also assume that S_C is the set of N_C cases from S_R that are solved by OK-SOLUTION and S_A (containing N_A cases) is the set from S_R - S_C that were solved by adaptation. Let also assume that the set S_{M1} contains N_{M1} cases out of S_K - S_R that can be solved by the MBR module within the prescribed time limits, whereas S_{M2} is the set of N_{M2} cases out of S_R - S_C - S_A that can be solved by invoking the MBR module (more specifically, MBR_2 since it is invoked just when the cases cannot be solved neither by solution replay nor adaptation). The parameters can then be estimated as reported in Table 6.

The (expected) costs are interpreted as an average time of performing the given task. For example, the expected cost of RETRIEVE is the average time spent by RETRIEVE in trying to retrieve the N_K cases of the test set S_K from the case memory. It is worth noting that both cost and probability of a given component (e.g. RETRIEVE) depends on a number of parameters characterizing that component (e.g. the S threshold on adaptability effort used in RETRIEVE).

In order to evaluate the contribution of the different modules constituting ADAPtER, let us examine the values of parameters estimated on test sets IS_3, IS_4 and CE reported in Table 7. It is worth noting that for many parameters, the results are quite similar for test sets IS_3 and IS_4 (both of them concern diagnostic problems from the domain D1), whilst the results for the test set CE are different (as expected, in particular for the computational cost).

In the three test sets the RETRIEVE module is quite effective since its computational cost is low, thanks to the retrieval strategies described in [36] and to the ability of the learning component to control the size of the case memory, by taking the swamping problem under

Parameter	Value		
	IS_3	IS_4	CE
Ps(RETRIEVE)	0.971	0.986	0.980
Pf(RETRIEVE)	0.029	0.014	0.020
Ps(OK-SOLUTION)	0.23	0.317	0.759
Pf(OK-SOLUTION)	0.77	0.683	0.241
Ps(ADAPTATION)	0.89	0.876	0.888
Pf(ADAPTATION)	0.11	0.124	0.112
Ps(MBR_1)	0.583	0.605	0.902
Ps(MBR_2)	0.941	0.916	1.000
Cost(MBR_1)	31339.0 msec.	30104.0 msec.	7630.7 msec.
Cost(MBR_2)	5766.0 msec.	7874.0 msec.	1778.7 msec.
Cost(RETRIEVE)	83.6 msec.	92.1 msec.	24.6 msec.
Cost(OK-SOLUTION)	18.8 msec.	16.3 msec.	6.8 msec.
Cost(ADAPTATION)	53.9 msec.	61.6 msec.	15.6 msec.

Table 7: Values of parameters of the cost model of ADAPtER for test sets IS_3, IS_4 and CE

control. We can notice that the probability of success for RETRIEVE is quite high, by taking into consideration that the system starts from an empty case memory.

As predicted by the theoretical results, the **OK-SOLUTION** module is inexpensive from the computational point of view (in fact, it essentially implements DIAGCHECK) and it is able to directly solve the input problem in some situations in the domain D1 (test sets IS_3 and IS_4). Moreover, **OK-SOLUTION** is useful since it focus the work of **ADAPTATION** by singling out the discrepancies between the replayed solution and the actual case to be solved. In the car fault domain (test set CE) **OK-SOLUTION** is really effective and in most cases there is no need for invoking the adaptation module since the re-play of the retrieved solution is sufficient to solve the new diagnostic problem.

ADAPTATION is quite effective since it is able to adapt a large majority of the cases submitted to this module. More important, the computational cost is quite low and the theoretical prediction that **ADAPTATION** is a costly process (see previous section) does not occur in practice. This excellent result is reached via the adoption of an approach for retrieving cases that is based on the estimation of the adaptation effort rather than on surface similarities between case. Moreover, the use of a suitable threshold S in **RETRIEVE** guarantees that the retrieved cases are not too difficult to be adapted.

As expected by theoretical results, the computational cost of invoking the MBR component is high, in particular as concerns **MBR_1**. The set of cases for which **RETRIEVE** fails to find a suitable easily adaptable case are hard to solve. In fact, in average **MBR_1** spends a significant amount of time in trying to solve them and the probability that the MBR reaches the time threshold without solving it is not negligible at all in the domain D1 (test sets IS_3 and IS_4).

The adoption of a cost model allows one to reason on the relative merits of the different

modules on the total performance on a multi-modal system [12]. Depending on the particular aspect of performance one is interest in, one could design alternative problem-solving architectures which combine basic problem solving modules in a different way or eliminate some of them.

If one puts a lot of emphasis in reducing the computational cost as much as possible, the cost model can single out that the module which has most impact on this parameter is the MBR module; as we have already seen in Table 7, both MBR_1 and MBR_2 have an average computational cost much larger that the other modules. So one could conceive an alternative architecture derived by ADAPtER where the MBR module is never activated. The resulting architecture (let us call it CBR for reasons discussed below) is the following

```
CBR(new-case,Case-Memory,Behavioral-Model,S,T1):

IF NOT RETRIEVE(input: new-case, Case-Memory, S, T1
                output: retrieve-solutions, T2)
   THEN return("failure")
   ELSE
    IF OK-SOLUTION(input: new-case, retrieve-solutions, Behavioral-Model, T2
                   output: replayed-solutions, T3)
     THEN return(replayed-solutions)
     ELSE
      IF ADAPTATION(input: new-case, replayed-solutions, Behavioral-Model, T3
                    output: adaptation-solutions, T4))
         THEN return(adaptation-solutions)
         ELSE return("failure")
```

It is easy to see that the new architecture contains only problem solving methods typical of the CBR approach (retrieval, replay solution, adaptation) and assumes that a problem cannot be solved if retrieval fails or adaptation fails, since the MBR component is not invoked as a backup problem solving method. In this way it would be possible to optimize computational time but the competence can be severely affected, since Table 7 shows that `Pf(ADAPTATION)` is not negligible and `Ps(MBR_2)` is relatively large in all the three test sets. More important, the CBR architecture has no mechanism for filling its gap of competence. In ADAPtER each time a problem is solved via MBR, there is an opportunity for learning, since the solved case is a candidate for learning; in this way, ADAPtER can start from an empty case memory and progressively developing and enhancing its competence via learning suitable cases solved via MBR. CBR cannot start from an empty case memory and therefore requires a training phase where suitable cases with their solutions are provided. The solution of a priori training phase is far from optimum, because it works just when there is a high probability that the characteristics of the diagnostic problems the system has to solve are steady over time.

In conclusion, the CBR architecture has been suggested as a result of cost analysis performed via the use of a cost model. However, its potential inadequacy as a general problem solver method has been singled out by considering the competence aspect (not only the static competence, but also the dynamics of competence via learning).

[12] Obviously, the utility of an analytical model depends on the accuracy of the prediction than can be made by using the model itself. The reader can easily verify that a very good correspondence exists between the measured cost of the ADAPtER architecture with the predicted value obtained by using cost model. For example, the measured average value of computational cost of ADAPtER on the test IS_3 set is 1525.8 (and the confidence interval at 95% is ± 322) whereas the analytical cost model provides an estimate of 1527.1 (by using the data reported in Table 7).

In the following we will make a further step in the multi-modal approach. Instead of analyzing different architectures (with different form of integration between CBR and MBR) in order to single out the most suitable architecture for a given domain, we will discuss how to select one of different architectures depending on the characteristics of the diagnostic problem to be solved, in order to select the most appropriate method for each single problem to be solved.

7 An Opportunistic Strategy for a Flexible Integration of CBR and MBR

The results discussed in the previous sections show the advantages (both in terms of competence and computational cost) of adopting a multi-modal approach instead of a pure MBR approach. However, the architecture of ADAPtER does not fully exploit the potential for improving the performance over MBR, since it does not take into account the characteristics of the cases included in the test set. The performance of ADAPtER are much better than the one of MBR in average, but it could be the case that in a number of cases MBR is better than ADAPtER not only in quality, but also in computational cost as already pointed in out in the very preliminary experiments reported in [33].

By analyzing the data reported in Table 2 it is quite clear that the amount of time consumed by MBR for solving a diagnostic problem can vary at a large extent also for cases which are apparently quite similar. The very large dispersion of computational cost is one of the main reasons why competence of MBR is not very high: there is a significant fraction of the diagnostic problems which are hard to solve (in our experiment they require more than 60 seconds of CPU time). The results presented so far show that the re-use of past experience is very useful, since many of the problems unsolvable by MBR (within the given time threshold) can be solved by re-playing or adapting solutions of the retrieved case. These diagnostic problems can be labeled as "hard" (at least from the MBR point of view) and for such kind of problems the control strategy of the CBR architecture should be more suitable than the one of ADAPtER, since no computational effort is spent for the MBR component, which would fail to solve the "hard" problem within the time limit.

However, the distribution of CPU time for test set IS_1 (see Table 2) shows very clearly that there is a significant amount of cases which can be solved by MBR with very limited computational resources. This kind of distribution is not peculiar of this test set or of this domain, since results obtained in other domains are consistent with this observation [33, 34].

This means that a significant amount of diagnostic problems are "easy" (from the MBR perspective) and the benefit of solving "easy" cases by means of CBR rather than MBR is highly questionable. In fact, there is a high chance that the CBR component does not save any time in solving the problem; on the contrary, there is the potential drawback that the diagnostic solutions provided by CBR are not exactly the ones given by MBR and therefore the quality of the solutions is lower.

The different merits of different problem solving architectures clearly depend on the typology of diagnostic cases which have to be solved. This suggests an alternative approach, where the particular form of integration of CBR and MBR is specific for the each type of diagnostic problem that has to be solved.

In the following we present a flexible architecture where an oracle is invoked to predict the difficulty of the problem at hand and a different problem solving architecture is invoked

according to the prediction of the oracle. In particular, we are interested in improving as much as possible the average computational cost, without loosing too much in competence and in quality (possibly improving also these parameters). For this reason we have identified three different classes of diagnostic problems (i.e. *easy*, *medium* and *hard* classes) depending on the actual effort needed by the MBR in solving them. The three classes are defined as:

$$Class(DP) = \begin{cases} easy & \text{if } MBR\_time(DP) \leq \tau_1 \\ medium & \text{if } \tau_1 < MBR\_time(DP) < \tau_2 \\ hard & \text{if } MBR\_time(DP) \geq \tau_2 \end{cases}$$

The threshold values in such a classification are somewhat arbitrary, depending on the computational power of the computer used for running the diagnostic system as well as the time requirements for the specific domain of application. In our experiments the thresholds for the classification are set to $\tau_1 = 1$ *sec.* and $\tau_2 = 1$ *min* [13].

The above characterization of the classes is based on the actual computation effort (i.e. $MBR\_time(DP)$) needed for solving the case. If we want to exploit a classification for deciding which problem solver architecture has to be invoked for solving the problem at hand, it is obvious that we need a classification mechanism able to classify the problems without actually solving them (or attempting to solve them). For this reason, we have to resort to an oracle able to estimate the class of computational effort of each problem by just inspecting the diagnostic problem description. While the actual definition of the oracle is reported in Section 7.1, the control strategy of the SUPERVISOR in the flexible architecture can be captured by the following rules:

```
IF oracle(DP) = easy THEN MBR(DP,T1)
IF oracle(DP) = medium THEN ADAPtER(DP,Case-Memory,Behavioral-Model,S,T1)
IF oracle(DP) = hard THEN CBR(new-case,Case-Memory,Behavioral-Model,S,T1)
```

The SUPERVISOR is not only responsible for selecting the problem solving method, but also for deciding under which conditions learning occurs. In particular, the solution of a diagnostic problem is learnt just in two cases:

```
IF oracle(DP) = medium and MBR(DP, Behavioral-Model, T1, mbr-solutions)
   THEN LEARN(mbr-solution, Case-memory)
IF oracle(DP) = easy and MBR(DP, Behavioral-Model, T1, mbr-solutions)
   and MBR-time > τ1 THEN LEARN(mbr-solutions, Case-memory)
```

It is worth noting that the new flexible multi-modal architecture assumes that *easy* cases are better to solve via MBR only. More important, we assume that there is a little computational advantage to cache solutions of the *easy* cases, so *easy* cases are not learnt, unless the supervisor

[13] It is worth noting that these thresholds partition the set of cases into not empty classes. For example, if we refer to Table 2 in the test set $IS2$ we have that 81.65% of the problems are *easy* (that is MBR is able to solve them in less than 1 sec of CPU time); even if we would adopt a more restrictive definition of *easy*, for example that *easy* problems are just the ones that can be solved within 200 msec., we can see that the class *easy* still contains 54% of the problems in $IS2$. It is worth noting that the class of *hard* problem is not empty in any of the training set reported in Table 2. Depending on the complexity of the domain and on the number of faults, the number of *hard* problems could be large (for example 12.9% in $IS1$) since all the cases that leaded to a memory overflow within the 1 hour time out required more than 1 min. of CPU time as well.).

discovers that the time taken by the MBR to solve them exceeds the time limit[14]. In this way, if the oracle has made a wrong prediction and the case is not actually a *easy* one, the learning mechanism is activated in order to include the solved case into the case memory and to fill a gap in its competence.

If a problem is classified as *medium*, there is reason to believe that MBR will take a significant amount of time for solving it, so it would be preferable to attempt to solve it via CBR and to resort to MBR only in case CBR fails. The cases classified as *medium* that cannot be solved via **OK-SOLUTION** or **ADAPTATION**, but have been solved by the MBR, are interesting cases to learn. One the meta-rule governing the behavior of the SUPERVISOR activates the learning mechanism when this situation occurs. In this way the case memory incrementally learns cases that are not straightforward to be solved by the MBR and therefore the system has the potential of saving computational time when a new cases similar to the learnt ones will be encountered.

The diagnostic problems classified by the oracle as *hard* are submitted just to the CBR component and no attempt is done for solving them via MBR even when CBR fails to solve them. More important, the solution of *hard* cases depend on the ability of adapting the solutions of *medium* cases (the only ones that are learnt by the flexible architecture) to become the solutions of *hard* cases.

7.1 Defining the Oracle

As stated above, the task of the oracle is to qualitatively estimate, for each diagnostic problem (and without attempting to solve it), the computational effort that would be needed to solve that problem by MBR. This estimation is performed by predicting (by means of a heuristic function $h_{MBR}(DP)$) the class (*easy*, *medium* or *hard*) to which the problem belongs.

Different heuristic functions could be defined by trying to correlate the MBR time with some parameters that can be easily evaluated by inspecting the specific problem to be solved.

In [28] we showed that in an abductive approach to diagnostic problem solving, the computational effort of solving a diagnostic problem is strongly correlated with the number of assignments $Cov_i \subseteq HYP$, containing only ground atoms representing abnormal behavioral modes, such that $BM \cup CXT \cup Cov_i \vdash \Psi^+$ for the diagnostic problem $DP = \langle BM, HYP, CXT, \langle \Psi^+, \Psi^- \rangle \rangle$.

In appendix B a particular heuristic function is introduced and discussed which tries to estimate the number of different global coverings Cov_i. Given such a heuristic $h_{MBR}(DP)$ function (provided that its values have a good correlation with the MBR time) the oracle can be defined in the following way:

$$Oracle(DP) = \begin{cases} easy & \text{if } h_{MBR}(DP) \leq S_1 \\ medium & \text{if } S_1 < h_{MBR}(DP) < S_2 \\ hard & \text{if } h_{MBR}(DP) \geq S_2 \end{cases}$$

where the thresholds S_1 and S_2 have to be learnt from experimental data (see appendix B.1 for details).

In order to test the accuracy of the $Oracle(DP)$ function, we have performed the following set of experiments. The training sets $IS1$ and $IS2$ have been used for learning the thresholds for domain $D1$. Similarly, CE has been used as training set for domain $D2$. In particular, for domain $D1$ we got $S_1 = 400$ and $S_2 = 12000$, while for domain $D2$ we got $S_1 = 1200$ and

[14]The time limit is represented by threshold τ_1, equal to 1 sec. in our experimental setting.

		easy	medium	hard
$IS3$	easy	82.52%	17.48%	0.0%
	medium	4.06%	95.54%	0.40%
	hard	0.0%	0.84%	99.16%
$IS4$	easy	83.14%	16.86%	0.0%
	medium	6.45%	89.34%	4.21%
	hard	0.0%	1.42%	98.58%
CE	easy	97.88%	2.12%	0.0%
	medium	5.28%	92.61%	2.11%
	hard	0.0%	0.0%	100%

Table 8: Classification results

$S_2 = 70000$. We have made use of the learnt thresholds on the test sets $IS3$ and $IS4$ for domain $D1$ (note that $IS3$ and $IS4$ have different characteristics with respect to the training sets $IS1$ and $IS2$) and CE itself for domain $D2$. Table 8 reports the classification results obtained by comparing the actual classes are (in the rows), with the estimated ones (in the columns). For each test set, the entry (X, Y) represents the percentage, over the set of problems of type X, of problems of type X classified as Y.

For example, in the test set $IS3$, 4.06% of the *medium* problems (according to the definition of $Class$) have been classified by $oracle(DP)$ as *easy*, whereas 95.54% have been correctly classified as *medium* and only 0.40% have been misclassified as *hard*.

The results reported in Table 8 show that the misclassification error is small and, more important, no *easy* problem is classified as *hard* and vice versa. So, we can claim that $h_{MBR}(DP)$ is sufficiently precise to be actually used for predicting the type of problem (the experimental results reported in the next section support this claim).

Furthermore, by pre-compiling some pieces of knowledge, the time spent in the computation of $h_{MBR}(DP)$ can be kept fairly small (see appendix B.).

7.2 Experimental Results

The effectiveness of the flexible integration can be fully appreciated by comparing its performance with the performance of the MBR problem solver and of the ADAPtER. Table 9 reports the results concerning computational cost, in particular the mean CPU time (in msec.) for solving diagnostic problems of different test sets together with its 95% confidence interval. It is easy to see that the flexible architecture presents a gain of about one order of magnitude over the MBR problem solver for diagnostic problems of domain D1 (test sets $IS3$ and $IS4$). There is also a quite significant saving in computational time of the flexible architecture with respect to ADAPtER. Most of the gain is obtained in the way *hard* problems are dealt with: only CBR is invoked for solving them and no attempt to resort to MBR in case of failure is foreseen. This strategy is very effective since not only allows one to save significant amount of computational resources but has almost no impact on the competence level with respect to ADAPtER (see Table 10), whilst there is a significant gain with respect to the practical competence of MBR.

This result is not obvious at all, since it shows that a limited number of cases of medium

	$IS3$	$IS4$	CE
MBR	4269.7 ±516.8	6468.8 ±580.6	1789.3 ±304.1
ADAPtER	1525.8 ±321.7	1258.7 ±260.6	241.3 ±131.6
Flex. arch.	440.8 ±81.9	466.8 ±79.6	229.0 ±42.2

Table 9: Comparison among the architectures: 95% confidence intervals for the average CPU time.

	$IS3$	$IS4$	CE
MBR	95.32%	92.90%	99.10%
ADAPtER	98.32%	98.73%	99.80%
Flexible arch.	99.28%	98.23%	99.50%

Table 10: Comparison among the architectures: percentage of solved problems (competence).

complexity is able to solve a significant portion of the *hard* diagnostic problems. In fact, it is worth remembering that the only cases that are learnt are the ones that are classified as *medium* and whose solution is provided by the MBR (after the failure of CBR). If we take into consideration that we start from an empty case memory and that the control strategy of the learning component decides to forget cases when they are considered no more useful (see [35] for details), the adequacy of the learning strategies and of the **ADAPTATION** mechanisms are evident.

A more detailed analysis of the competence of the different modules composing the flexible architecture confirms the claim. Let us consider the test set $IS4$. The oracle classified 1772 out of the 3000 cases composing $IS4$ (59.07%) as *easy*. All of them are solved by the MBR and only for 46 cases (2.6%) the prediction of oracle was too optimistic. In these 46 cases the MBR architecture was able to solve the diagnostic problems, but the CPU time needed for solving each of them exceeded the time threshold of 1 sec. According to the meta-rules governing the learning process these 46 cases were learnt and added to the case memory. Oracle classified 990 cases as *medium*; all of them were solved within the time threshold of 1 minute by ADAPtER. More precisely, 843 cases out of the 990 (85.15%) were solved by means of **OK-SOLUTION** and **ADAPTATION** and for 147 cases out of the 990 (14.85%)the MBR was successfully invoked to solve the problems after the failure of the adaptation (or retrieval). It is worth noting that the 147 cases were learnt and added to the case memory. The oracle classified 238 cases of the $IS4$ test set as *hard* (7.93% of the cases in $IS4$). By using just the CBR architecture 185 out of the 238 have been solved, whilst in 53 case either the retrieval or the adaptation mechanism failed. This has to be compared with the result of the pure MBR architecture where 213 out the 3000 cases in $IS4$ were not been solved.

As regards $IS3$, Table 10 shows that the competence of the flexible architecture is greater than the competence of ADAPtER. This could be a bit surprising, since ADAPtER always

	$IS3$	$IS4$	CE
ADAPtER	0.675	0.573	0.426
Flexible arch.	0.882	0.804	0.888

Table 11: Comparison among the architectures: the quality parameter.

has access to both reasoning methods (CBR and MBR). However, this result is explained by the different learning process in the two architectures (and therefore by the possibly different content of the two case memories after the same set of input cases have been submitted to the two diagnostic systems). Indeed, the problems in $IS3$ that either ADAPtER or the flexible arcitecture was not able to solve were all *hard* problems; however, the set of cases learnt by the flexible architecture allowed it to solve by CBR some *hard* problems that ADAPtER was not able to solve. In particular, even if the flexible architecture did not solve 4 *hard* problems of $IS3$ that ADAPtER did solve, it solved (by CBR) 28 *hard* problems that ADAPtER was unable to solve.

If we analyze the results for test set CE (diagnostic cases from domain D2 - car faults), we see a significant gain of the flexible architecture with respect the pure MBR architecture. However, there is no big difference in computational cost (and competence) if we compare the flexible architecture and ADAPtER. This result is not surprising because domain D2 is not as complex as domain D1. In fact, the number of *hard* cases in test set CE is relatively small and therefore the saving that can be obtained via adopting the flexible architecture (which does not invokes MBR in case of failure of the CBR for *hard* cases) concerns just a small fraction of the whole test set. However, the adoption of the flexible architecture is quite beneficial for another performance parameter: the quality. By inspecting Table 11 it is easy to see that in all test sets the quality improves significantly with respect to ADAPtER. This is not surprising since the flexible architecture has been designed with such a goal in mind: the decision that *easy* problems have to be solved by MBR guarantee that all minimal diagnostic solutions are got and therefore the quality is optimum. The amount of increase in the quality is really significant and allows to conclude that the flexible architecture provides a real improvement over the ADAPtER architecture in all the test sets that have been analyzed.

8 Discussion and Related Works

The integration of different reasoning modes has been extensively investigated by several researchers and, as already mentioned in the introduction, the use of the CBR paradigm has received particular attention, because of the availability of solved (often by human experts) cases in several problem solving tasks. The interest in understanding how far one can go with the CBR approach in problem solving also arises because of the basic assumptions the CBR approach is based on. In fact, a pure case-based problem solver may suffer from the same drawbacks of a pure "shallow" reasoning mode like rule-based reasoning, since the competence of a CBR problem solver depends on the content of the case base; the case base must be sufficiently complete in order to avoid knowledge gaps. Moreover, adaptation often require to devise a restricted rule-based reasoner for that task [23]. On the contrary, CBR can be seen as complementary to the use of a "deep" reasoning mode like model-based or constraint-based rea-

soning where background knowledge is always used from scratch and no learning by experience is usually provided.

The integration of CBR with other reasoning modes can be exploited in two different ways: at the *domain level* for providing alternative way of solving a given problem, possibly by choosing the best mode or by using one mode to refine the others; at the *internal reasoning level*, by providing a way of using one mode for guiding the internal reasoning process provided by another mode. Several examples are present in both categories; in the first class we can find systems that use CBR for guiding the application of ill-defined or very general rules in legal domains [10, 39] or in medical ones [8, 9], for breaking decisions where rules conflict [4], for selective selection of reasoning modes in path planning [21], for integrating schema-based and model-based design in very complex tasks [7], for helping decision making in a Bayesian framework [11] or for trying to solve problems that are exceptions to rules [45] or prototypes [46] in classification tasks.

The approaches mentioned above are mainly concerned with the improvement of competence and quality of solutions, since multiple reasoning paradigms are used to compensate lack of knowledge of single ones. However, the integration of CBR at the domain level can be also triggered by computational complexity issues as in CASEY [24] where model-based results are captured into cases in order to speed-up problem solving.

The second class of approaches refers to the use of CBR as an internal resource of the inference engine of the system, since CBR is integrated at the reasoning level; relevant examples are the PRODIGY-ANALOGY planner [50] where the reasoning process is cached for exploiting similar reasoning traces and the DIAL planning system [25] where transformational CBR generates a new plan by adapting prior plans. Plan adaptation is initially provided by rule-based reasoning, but internal derivational CBR captures the adaptation process, in order to supplant rule-based adaptation and similarity assessment.

Our approach exhibits features of both categories: it provides a flexible, opportunistic and adaptive integration of CBR and MBR at the domain level, by addressing every aspect of performance namely computational complexity, competence and solution quality; moreover it also provides integration at the reasoning level both "internal" and "external" to the problem solving cycle: internal, since adaptation is performed as a focused step of model-based reasoning exploiting the derivational trace of the retrieved solution, external since the *oracle* is able to suggest the best reasoning style for each proposed case to be solved. In particular, the use of a "solution-replay" mechanism seems to provide significant advantages in practical terms, to the classical REUSE step of the CBR cycle. This is not peculiar to the diagnostic task, since important complexity results have also been obtained in planning systems exploiting Derivational Analogy whose fundamental step is indeed the solution-replay. In [5], the authors present **DerUCP**, a general model for plan adaptation using Derivational Analogy. They show that the general complexity of case-based planners ([31]) can be significantly mitigated if a Derivational Analogy framework is adopted. Their results seem then to furtherly confirm the performance results practically obtained by our multi-modal architecture. Indeed, ADAPtER presents some similarities with DerUCP; the *replay step* in DerUCP is conceptually equivalent to the replay of the retrieved solutions performed by the SOLUTION RE-PLAYER in ADAPtER, the backtracking of nodes in the replay path in DerUCP corresponds to *Inconsistency Removal Step* in ADAPtER, furthermore, both DerUCP (in what is called *Extension* step) and ADAPtER (in its *Explanation Construction* step) make use of first principle reasoning to complete their task.

However, despite these similarities there are also two important differences. First of all, each case in ADAPtER stores the solutions of a past problem. Differently, in DerUCP, "the cases

contain the sequence of decisions made to obtain a plan rather than the plan itself" [5]; however, this difference is lowered by the fact that both such objects are used as derivational traces for the solution-replay.

A second difference concerns the fact that in DerUCP, solving a planning problem by adapting an old plan never requires a computational effort greater than the computational effort that would be needed to solve the problem from scratch, if the adaptation is performed via Derivational Analogy. It is worth pointing out that ADAPtER, in general, does not guarantee such a property. Indeed, it may happen that ADAPtER solves a problem from scratch (i.e. by MBR) after having failed in the attempt of adapting a solution. In such a case, ADAPtER actually performs the work needed to solve the problem from scratch plus the work needed to adapt a solution (and therefore it performs an unfruitful search in the *Explanation Construction* step). Nevertheless, we have to notice that whenever **OK-SOLUTION** succeeds the problem is solved with no search at all in the behavioral model. Moreover, in those cases in which **ADAPTATION** succeeds, search is performed only by the *Explanation Construction* step which does not explore a portion of the search space larger than the portion that would be explored by MBR. Experimental results (and the analogies with DerUCP) lead to argument that in many cases, the portion of the state space that is explored is significanlty smaller than the one that would have been explored from scratch.

It should be clear that the capability of retrieving cases whose solutions have a good chance of being immediately re-usable or successfully adapted to the new problem plays an important role in our architecture. As mentioned above, the function of match used by **RETRIEVE** does not simply measure the surface similarity between cases. Instead, it estimates the computational effort that would be needed to adapt a solution to a stored case, in order to solve the input one. This retrieval mechanism proved to be rather effective in retrieving solutions with a high probability of being successfully (and efficiently) adapted (see Section 5).

Finally, one important contribution of the paper concerns the definition of the *oracle* of Section 7. In fact, its adoption is not only useful for improving the different aspects of performance as shown in Section 7.2, but changes the way the integration between different problem solving methods is conceived. In fact, without the oracle a fixed straight-line integration like the *master/slave* one of ADAPtER may result adequate for a given mix of problems, but inadequate for a different mix. Let us suppose that for some reason the diagnostic system has to deal just with very hard cases: MBR could result inappropriate because its practical competence could be very low (as shown in the experiments, many hard cases are not solvable within a reasonable time limit). On the contrary, if the diagnostic system has to deal with simple problems (such as single fault diagnostic problems) CBR may not be the most appropriate approach, since there is no improvement in computation time and a degradation in the quality of solution with respect to the solutions provided by MBR.

The ability of defining an accurate oracle paves the way for having an efficient and competent problem solver independently from the particular characteristics of the set of problems submitted for solution. It is up to the oracle to select the most appropriate sequence of invocation of problem solvers for the particular problem at hand, transforming a rigid master/slave architecture in a more inter-wined one (a more *collaborative integration* using the terminology of [29]). Obviously, this kind of adaptive behavior has to be coupled with the adaptive capabilities of the case memory management system; in fact, the content itself of the case memory should change over time for adapting its competence to the cases the system has to deal with. Fortunately such kind of case memory management systems exist (see [26]) and we have developed learning

and forgetting techniques that are resulted very effective [35])

The paper has investigated the multi-modal reasoning approach in the framework of diagnostic problem solving. The results (both on the theoretical side and the experimental one) hold for such a task. However, the methodology we have devised for the analysis of multi-modal reasoning is suitable for many other tasks. In particular, a theoretical analysis can show whether the re-use of past solutions is always more efficient than solving the problem from scratch. If not, there is the need for singling out what kinds of integration between CBR approach and "first-principles" problem solving are possible and convenient. It is worth noting that the "utility problem" has be dealt with, so it is not possible to conceive an integrated system where the case memory continues just to grow and all cases that are solved are learnt. Strategies based on models of competence and/or of usefulness have to guide the learning process. We have shown that the use of analytic methods (such as cost models) provides significant insight in understanding the pros and cons of the different problem solving methods and may suggest new strategies for multi-modal reasoning.

As reported in [29], the main open issue in CBR integrations (but we can say in multi-modal reasoning in general) is to evaluate advantages and disadvantages of different architectures through a deep experimental and analytical evaluation: we believe that our work is a step in this direction.

Acknowledgments

We thank M. van Someren for many useful discussions on the role of cost models in analyzing multi-modal reasoning systems. We also want the thank the anonymous referees for the very useful suggestions that really helped in improving the paper.

A Proofs of Theorems

A.1 Proof of theorem 3.1

First we prove that DIAGSAT is NP-hard by means of a reduction from SAT (the problem of deciding the satisfiability of a boolean formula in CNF) that is known to be NP-complete [6]. Let ϕ be a boolean formula in CNF with variables $V = \{v_1, v_2, \ldots v_n\}$ and clauses $C = \{c_1, c_2, \ldots c_m\}$. An assignment of truth values to every variable in V is called a *variable assignment*. We will construct a diagnostic problem $DP = \langle BM, HYP, CXT, \langle \Psi^+, \Psi^- \rangle \rangle$ such that DP has a solution if and only if ϕ is satisfiable[15]. Without restriction, let us consider a propositional model BM using the following propositional letters:

* T_i $(1 \leq i \leq n)$ meaning that v_i=true has been selected,
* F_i $(1 \leq i \leq n)$ meaning that v_i=false has been selected,
* S_i $(1 \leq i \leq n)$ meaning that a truth value for v_i has been selected,
* C_j $(1 \leq j \leq m)$ meaning that c_j is satisfied.
* $\overline{C_j}$ $(1 \leq j \leq m)$ meaning that c_j is not satisfied.

[15] We use the approach proposed for the proof of Theorem 4 in [31] as a guideline.

The model BM is composed by the following clauses:

* $T_i \rightarrow S_i$ $(1 \leq i \leq n)$

* $F_i \rightarrow S_i$ $(1 \leq i \leq n)$

* $T_i \wedge S_1 \wedge \ldots S_n \rightarrow C_j$ if $v_i \in c_j$

* $F_i \wedge S_1 \wedge \ldots S_n \rightarrow C_j$ if $\overline{v_i} \in c_j$

* $T_{i_1} \wedge \ldots T_{i_k} \wedge F_{j_1} \wedge \ldots F_{j_h} \wedge S_1 \wedge \ldots S_n \rightarrow \overline{C_j}$
  if $c_j = v_{j_1} \vee \ldots v_{j_h} \vee \overline{v_{i_1}} \vee \ldots \overline{v_{i_k}}$

We also have the following meta-level constraints:
$T_i \wedge F_i \rightarrow \bot \qquad C_j \wedge \overline{C_j} \rightarrow \bot$
We further assume $HYP = \{T_i\} \cup \{F_i\}$, $CXT = \emptyset$, $\Psi^+ = \{C_1, \ldots C_m\}$ and $\Psi^- = \{\overline{C_1}, \ldots \overline{C}_m\}$.

Notice that the last three clauses of the model BM are mutually exclusive; they model the fact that C_j is satisfied (third and fourth clause) or not satisfied (fifth clause). Indeed, every assignment $H \subseteq HYP$ such that $BM \cup CXT \cup H \vdash C_j$ $(1 \leq j \leq m)$ is such that $BM \cup CXT \cup H \nvdash \overline{C_j}$ $(1 \leq j \leq m)$. Moreover, every assignment $H \subseteq HYP$ and such that $\mathcal{P}(H) = \mathcal{P}(HYP)$ corresponds to a variable assignment and vice versa.

If ϕ is satisfiable, the set $H \subseteq HYP$ corresponding to the satisfying variable assignment is a diagnosis for DP; indeed, since every c_j is satisfied, then every C_j will be derived from H and consequently no $\overline{C_j}$ will be derived from H. Conversely, if DP has a solution, let H be a total diagnosis for DP, then the variable assignment in V correspondent to H clearly satisfies ϕ.

Finally, DIAGSAT is in NP since the following is a non deterministic algorithm running in polynomial time (see proposition 3.1):

- guess a solution H to DP;

- DIAGCHECK on H and DP.

A.2 Proof of Theorem 3.2

Dim. To prove that DASAT is NP-complete, we must prove the following properties:

1. DASAT is in NP;

2. DASAT is NP-hard.

1. The first property can be easily proved. Let's consider a diagnostic problem $DP_1 = \langle BM, HYP, CXT_1, \langle \Psi_1^+, \Psi_1^- \rangle \rangle$, a diagnosis H for $DP = \langle BM, HYP, CXT, \langle \Psi^+, \Psi^- \rangle \rangle$ an integer $k \leq |H|$ and an assignment H_1 (representing the guess) such that $\mathcal{P}(H_1) = \mathcal{P}(HYP)$. To verify if H_1 is a diagnosis for DP_1 containing a sub-assignment of H of cardinality at least k we have to verify:

   (a) if H_1 contains at least k atoms occurring in H; this operation has complexity $O(r^2)$, where $r = card(\mathcal{P}(HYP))$;

(b) if H_1 is a diagnosis for DP_1. This is the DIAGCHECK problem which is in P (proposition 3.1).

2. To prove the NP-hardness of DASAT we make a reduction from DIAGSAT which is NP-complete (theorem 3.1)

   Let's consider the following instance of the problem DIAGSAT:

   $DP = \langle BM, HYP, CXT, \langle \Psi^+, \Psi^- \rangle \rangle$.

   DP can be reduced to an instance of DASAT as follows:

   $\overline{DP} = \langle \overline{BM}, \overline{HYP}, CXT, \langle \overline{\Psi}^+, \overline{\Psi}^- \rangle \rangle$, where:

   $\overline{BM} = BM \cup$
   $\{(\bigwedge_{m(a)\in\Psi^+} m(a)) \rightarrow m1\_new(new1),$
   $(\bigwedge_{m(a)\in\Psi^+} m(a)) \rightarrow m2\_new(new1),$
   $(\bigwedge_{p\in\mathcal{P}(HYP)} p(new)) \rightarrow m1\_new(new1)\} \cup$
   $(\bigcup_{p\in\mathcal{P}(HYP)} \{p(new) \rightarrow m2\_new(new2)\})$;

   $\overline{HYP} = HYP \cup \{p(new)/\ p \in \mathcal{P}(HYP)\}$;

   $\overline{\Psi}^+ = \{m1\_new(new1)\}$;

   $\overline{\Psi}^- = \{m1\_new(new2)\} \cup \Psi^-$,

   where the predicate symbols $m1\_new$, $m2\_new$ and the constants new, $new1$ and $new2$ do not belong to the language of BM.

   It is easy to see that $\overline{H} = \{p1(new), \ldots, pr(new)\}$ is a diagnosis for $\overline{DP}$. Indeed, $\overline{H}$ is an assignment for $\mathcal{P}(HYP)$ such that

   $\overline{BM} \cup CXT \cup \overline{H} \vdash m1\_new(new1)$

   $\overline{BM} \cup CXT \cup \overline{H} \nvdash m1\_new(new2)$

   $\forall n(a) \in \Psi^-,\ \overline{BM} \cup CXT \cup \overline{H} \nvdash n(a)$.

   Furthermore, no atom can be derived from $\overline{BM}$ alone, since neither BM nor $\overline{BM} - BM$ contain facts (i.e. definite clauses with an empty body; see definition 2.1). Moreover, the body B of each clause in BM does not contain any $p(new) \in \overline{H}$ and every $c(X) \in CXT$ can occur in B only in conjunction with some other atom $s(y) \in CXT$ (see definition 2.1), therefore the set of facts $CXT \cup \overline{H}$ does not satisfy the body of any clause in BM. All clauses in $\overline{BM}$ whose body is satisfied by $CXT \cup \overline{H}$ are in $\overline{BM} - BM$, but the head of these clauses are atoms not belonging to the language of BM, therefore they are different from each $n(a) \in \Psi^-$ and from each atom occurring in the clauses of BM. It follows that

   $\forall n(a) \in \Psi^-,\ \overline{BM} \cup CXT \cup \overline{H} \nvdash n(a)$.

   Let $\overline{DP}_1 = \langle \overline{BM}, \overline{HYP}, CXT, \langle \overline{\Psi}_1^+, \overline{\Psi}_1^- \rangle \rangle$, where

   $\overline{\Psi}_1^+ = \overline{\Psi}^+ \cup \{m2\_new(new1)\}$ and

   $\overline{\Psi}_1^- = \overline{\Psi}^- \cup \{m2\_new(new2)\}$

   and let's consider the instance of DASAT consisting in determining whether there exists a diagnosis $\overline{H}'$ for $\overline{DP}_1$ containing a sub-assignment of $\overline{H}$ of cardinality at least $k = 0$.

It should be clear that this instance of DASAT has been defined by means of a polynomial transformation of the instance of DIAGSAT relevant to DP.

To prove that DASAT is NP-complete we need only to prove that $\overline{H}'$ is a diagnosis for $\overline{DP}_1$ *iff* $\overline{H}'$ is a diagnosis for DP (having chosen $k = 0$, a diagnosis for $\overline{DP}_1$ is a solution to the considered instance of DASAT and vice versa).

Hp. $\overline{H}'$ is a diagnosis for $\overline{DP}_1$.

Ts. $\overline{H}'$ is a diagnosis for DP.

$\overline{H}'$ is a diagnosis for $\overline{DP}_1$, thus (def. 2.2) $\mathcal{P}(\overline{H}') = \mathcal{P}(\overline{HYP})$, therefore

(1) $\mathcal{P}(\overline{H}') = \mathcal{P}(HYP)$

(Indeed, given the definition of $\overline{HYP}$, $\mathcal{P}(\overline{HYP}) = \mathcal{P}(HYP)$). Moreover, given

$\overline{\Psi}_1^+ = \{m1\_new(new1), m2\_new(new1)\}$, we have

(a) $\overline{BM} \cup CXT \cup \overline{H}' \vdash m1\_new(new1)$ and

(b) $\overline{BM} \cup CXT \cup \overline{H}' \vdash m2\_new(new1)$

and given

$\overline{\Psi}_1^- = \{m1\_new(new2), m2\_new(new2)\} \cup \{n(a)/\ n(a) \in \Psi^-\}$,

(c) $\overline{BM} \cup CXT \cup \overline{H}' \nvdash m1\_new(new2)$

(d) $\overline{BM} \cup CXT \cup \overline{H}' \nvdash m2\_new(new2)$

(e) $\forall n(a) \in \Psi^- \overline{BM} \cup CXT \cup \overline{H}' \nvdash n(a)$.

We remark that $\overline{H}'$ does not contain any $pi(new) \in \overline{H}$ (i.e. the only sub-assignment of $\overline{H}$ contained in $\overline{H}'$ has cardinality $k = 0$). Indeed, if it was $pi(new) \in \overline{H}'$, we would have $\overline{BM} \cup CXT \cup \overline{H}' \vdash m2\_new(new2)$, since $\overline{BM}$ contains the implication $pi(new) \rightarrow m2\_new(new2)$. But this fact would contradict the statement d. So, it follows,

(2) $\overline{H}' \subseteq HYP$;

Furthermore, in $\overline{BM}$ only two implications have $m1\_new(new1)$ in their head, namely

$(\bigwedge_{m(a) \in \Psi^+} m(a)) \rightarrow m1\_new(new1)$ and

$(\bigwedge_{p \in \mathcal{P}(HYP)} p(new)) \rightarrow m1\_new(new1)$; since

$\overline{BM} \cup CXT \cup \overline{H}' \vdash m1\_new(new1)$ (statement a) and, as said above, for all $p \in \mathcal{P}(HYP)$, $p(new) \notin \overline{H}'$, therefore only the first implication can be used to derive $m1\_new(new1)$. This means that

$\forall m(a) \in \Psi^+ \overline{BM} \cup CXT \cup \overline{H}' \vdash m(a)$, but no implication added to BM in order to build $\overline{BM}$ has any $m(a) \in \Psi^+$ in its head, thus

(3) $\forall m(a) \in \Psi^+ BM \cup CXT \cup \overline{H}' \vdash m(a)$.

$BM \subset \overline{BM}$, therefore, from statement e, we have

(4) $\forall n(a) \in \Psi^- BM \cup CXT \cup \overline{H}' \nvdash n(a)$.

From (1)-(4) it follows that $\overline{H}'$ is a diagnosis for DP.

We still have to prove that each diagnosis for DP is a diagnosis for $\overline{DP}_1$.

Hp. $\overline{H}'$ is a diagnosis for DP;

Ts. $\overline{H}'$ is a diagnosis for $\overline{DP}_1$.

$\overline{H}'$ is a diagnosis for DP, thus $\mathcal{P}(\overline{H}') = \mathcal{P}(HYP)$, so

(5) $\mathcal{P}(\overline{H}') = \mathcal{P}(\overline{HYP})$ and

(6) $\overline{H}' \subseteq HYP \subset \overline{HYP}$.

$\forall m(a) \in \Psi^+$ $BM \cup CXT \cup \overline{H}' \vdash m(a)$, therefore $\forall m(a) \in \Psi^+$ $\overline{BM} \cup CXT \cup \overline{H}' \vdash m(a)$ (since $BM \subset \overline{BM}$), thus $\overline{BM} \cup CXT \cup \overline{H}' \vdash m1\_new(new1)$ e $\overline{BM} \cup CXT \cup \overline{H}' \vdash m2\_new(new1)$, i.e.

(7) $\forall x \in \overline{\Psi}_1^+$ $\overline{BM} \cup CXT \cup \overline{H}' \vdash x$.

Furthermore, $\forall n(a) \in \Psi^-$ $BM \cup CXT \cup \overline{H}' \nvdash n(a)$; the head of each clause added to BM in order to build $\overline{BM}$ contains only symbols that do not belong to the language of BM, thus $\forall n(a) \in \Psi^-$ $\overline{BM} \cup CXT \cup \overline{H}' \nvdash n(a)$. In other words,

(f) for those $n(a) \in \overline{\Psi}_1^-$ that belong to Ψ^-, we have $\overline{BM} \cup CXT \cup \overline{H}' \nvdash n(a)$.

Moreover, no clause in $\overline{BM}$ has $m1\_new(new2)$ in its head, thus

(g) $\overline{BM} \cup CXT \cup \overline{H}' \nvdash m1\_new(new2)$.

The only clauses in $\overline{BM}$ having $m2\_new(new2)$ in their head are $p(new) \rightarrow m2\_new(new2)$, for each $p \in \mathcal{P}(HYP)$, but no clause in $\overline{BM}$ has any $p(new)$ (with $p \in \mathcal{P}(HYP)$) in their head and $p(new) \notin \overline{H}'$ (since $\overline{H}' \subseteq HYP$ and $p(new) \notin HYP$), therefore

(h) $\overline{BM} \cup CXT \cup \overline{H}' \nvdash m2\_new(new2)$.

From (f)-(h) it follows

(8) $\forall x \in \overline{\Psi}_1^-$ $\overline{BM} \cup CXT \cup \overline{H}' \nvdash x$

From (5)-(8) it follows that $\overline{H}'$ is a diagnosis for $\overline{DP}_1$.

This proves that DASAT is NP-hard. Since DASAT has also been proved to be in NP, it follows that DASAT is NP-complete.

B Heuristic Function

In Section 7.1 an oracle has been defined which classifies each diagnostic problem DP into three classes (i.e. *easy*, *medium* and *hard* classes) depending on the estimated computational effort needed to solve DP by MBR. Such an oracle makes use of a heuristic function $h_{MBR}(DP)$ that we describe in this appendix.

Let $\mathcal{DP}$ be the set of all diagnostic problems. The function $h_{MBR} : \mathcal{DP} \rightarrow \mathcal{N}$ associates a natural number to each diagnostic problem DP $in \mathcal{DP}$. In order to be effective, $h_{MBR}(DP)$ has to be correlated with the actual time $MBR_{time}(DP)$ that would be needed to solve the problem DP by MBR: i.e. the greater $h_{MBR}(DP)$, the greater $MBR_{time}(DP)$; the smaller $h_{MBR}(DP)$, the smaller $MBR_{time}(DP)$.

To understand how such a function can be defined, it is necessary to understand how the MODEL-BASED REASONER works.

Let's recall that in the framework that we considered for the experiments each diagnostic problem $DP = \langle BM, HYP, CXT, \langle \Psi^+, \Psi^- \rangle \rangle$ is such that BM describes the faulty behavior of the system to be diagnosed (i.e. the clauses in BM describe the consequences, both direct and indirect, of the presence of faults in the components). The set OBS of observations are

partitioned into two sets $OBS = OBS^A \cup OBS^N$ of abnormality (OBS^A) and normality (OBS^N) observations (to actually have a diagnostic problem, it must be $OBS^A \neq \emptyset$). Since we are interested in diagnoses that *cover* (i.e. that explain abductively) all the abnormal observations, we put $\Psi^+ = OBS^A$. Ψ^- is computed on the basis of all the observations, therefore $\Psi^- = \{m(y)/y \; is \; an \; admissible \; value \; for \; the \; observable \; parameter \; m \; \wedge (\exists m(x))(m(x) \in OBS^A \cup OBS^N \wedge x \neq y)\}$.

Moreover, we impose a minimality criterion on diagnoses: given two diagnoses H_1 and H_2 for a diagnostic problem DP, if the set of faulty behavioral modes occurring in H_1 is a subset of those occurring in H_2, then H_1 is preferred w.r.t. H_2. Each diagnosis H such that no preferred diagnosis w.r.t. H exists is called a *minimal diagnosis*.

A MODEL-BASED REASONER that computes the set of the minimal diagnoses for each given diagnostic problem DP can achieve its task in the following basic steps:

1. **Find Coverings:** compute the set $MIN\_COVS_{DP}(OBS^A)$ of the sub-assignments $Cov \subset HYP$ to $\mathcal{P}(HYP)$ such that $BM \cup CXT \cup Cov \vdash \bigwedge_{m(x) \in OBS^A} m(x)$ and each $Cov \in MIN\_COVS_{DP}(OBS^A)$ is minimal w.r.t. set inclusion (i.e. there is no $Cov' \in MIN\_COVS_{DP}(OBS^A)$ such that $Cov' \subset Cov$); Cov is called a *minimal covering* for OBS^A.

2. **Filter Consistent:** discard all the coverings $Cov \in MIN\_COVS_{DP}(OBS^A)$ for which there is $m(y) \in \Psi^-$ such that $BM \cup CXT \cup Cov \vdash m(y)$. Let $CONS\_MIN\_COVS_{DP}(OBS^A)$ be set of the remaining coverings (i.e. those ones that are consistent with all the observed parameters).

3. **Complete:** for each $Cov \in CONS\_MIN\_COVS_{DP}(OBS^A)$, build a diagnosis $Diag(Cov) = Cov \cup (\bigcup_{p \in \mathcal{P}(HYP) - \mathcal{P}(Cov)} p(normal)$

The first step computes the set of the abductive explanations for the observed abnormal parameters (first condition in definition 2.2) and it retains only those explanations that are minimal w.r.t. set inclusion, whereas the second step discards all those explanations entailing some inconsistency w.r.t. the whole set of observations (second condition in definition 2.2). The third step simply completes each minimal covering Cov computed in the previous steps by stating the normal behavioral mode for each predicate symbol in $\mathcal{P}(HYP)$ and not occurring in $\mathcal{P}(Cov)$.

The *find coverings* step can be refined into four sub-steps. For the sake of clarity, we first describe these sub-steps without considering any mechanism aimed at focusing the reasoning process, as follows:

1.1 For each $m(a) \in OBS^A = \{m_1(a_1), \ldots, m_k(a_k)\}$, compute the set $MIN\_COVS_{DP}(\{m(a)\})$ of the minimal coverings for the singleton set of observations $\{m(a)\}$ (i.e. $MIN\_COVS_{DP}(\{m(a)\})$ is the set of the abductive explanations for the observation $m(a)$, minimal w.r.t. set inclusion);

1.2 compute the set

$S_{DP}(OBS^A) = \{\bigcup_{i=1}^{k} Cov_i / Cov_1 \in MIN\_COVS_{DP}(\{m_1(a_1)\}), \ldots,$

$Cov_k \in MIN\_COVS_{DP}(\{m_k(a_k)\})\}$ by taking each element $\langle Cov_1, \ldots, Cov_k \rangle$ in the Cartesian product

$MIN\_COVS_{DP}(\{m_1(a_1)\})\times,\ldots,\times MIN\_COVS_{DP}(\{m_k(a_k)\})$ and computing the union $Cov_1 \cup \ldots \cup Cov_k$;

1.3 compute the set $COVS_{DP}(OBS^A) = \{C \in S_{DP}(OBS^A)/C$ *is an assignment to* $\mathcal{P}(\mathcal{C})\}$, containing all the set of atoms $C \in S_{DP}(OBS^A)$ such that C does not contain two different ground instances $p(a)$ and $p(b)$ for a same predicate $p \in \mathcal{P}(C)$ (each $Cov \in COVS_{DP}(OBS^A)$ represents an abductive explanation for all the observations in OBS^A and it is called a *covering* for OBS^A);

1.4 restrict $COVS_{DP}(OBS^A)$ to the set $MIN\_COVS_{DP}(OBS^A)$ of the coverings for OBS^A minimal w.r.t. set inclusion.

As stated above, these four sub-steps describe an algorithm that does not make use of any mechanism to focus the search for the coverings $COVS_{DP}(OBS^A)$. Actually, the MODEL-BASED REASONER which we have taken into consideration in the present paper does make use of a focusing mechanism in order to avoid the computation of some inconsistent coverings. Such a mechanism is based on a set of necessary conditions that are pre-compiled from the behavioral model BM and thus they hold for each diagnostic problem relevant to BM. We refer to [15, 13] for the details about the pre-compilation techniques of such necessary conditions and their use in focusing the abductive reasoning. To understand the heuristic function $h_{MBR}(DP)$, it is sufficient to know that a set of necessary conditions $Nec(Cov)$ can be associated with each minimal covering $Cov \in MIN\_COVS_{DP}(\{m(a)\})$, for each $m(a)$, where m is an observable parameter and a is an admissible abnormal value for m. $Nec(Cov) = \{n_1(v_1), \ldots, n_h(v_h)\}$ is a set of ground atoms such that each n_i is an observable parameter and each v_i is an admissible abnormal value for n_i $(i = 1, \ldots, h)$. For each $Cov \in MIN\_COVS_{DP}(\{m(a)\})$ and each $n(v) \in Nec(Cov)$, it holds that $BM \cup Cov \vdash n(v)$.

If there are $m(a) \in OBS^A$ and $Cov \in MIN\_COVS_{DP}(\{m(a)\})$ such that $Nec(Cov) \cap \Psi^- \neq \emptyset$, then every $Cov' \in MIN\_COVS_{DP}(OBS^A)$ such that $Cov \subseteq Cov'$ would be discarded in the *filter consistent* step. By taking into account this fact, the MODEL-BASED REASONER can use these necessary conditions in sub-step 1.1 in order to avoid the computation of some inconsistent coverings that would be discarded in the *filter consistent* step. Indeed, the sub-step 1.1 actually computes, for each $m(a) \in OBS^A$, the subset $\overline{MIN\_COVS}_{DP}(\{m(a)\}, \Psi^-)$ of $MIN\_COVS_{DP}(\{m(a)\})$ containing all and only the minimal coverings Cov for $m(a)$ such that $Nec(Cov) \cap \Psi^- = \emptyset$. The computation of $COVS_{DP}(OBS^A)$ (sub-steps 1.2 and 1.3) is actually based on these $\overline{MIN\_COVS}_{DP}(\{m(a)\}, \Psi^-)$ sets instead of $MIN\_COVS_{DP}(\{m(a)\})$ (for each $m(a) \in OBS^A$).

In [28] it is shown experimentally that the computational effort of solving a diagnostic problem by MBR is correlated with the number of coverings for the setOBS^A of abnormality observations: the grater the cardinality of
$COVS_{DP}(OBS^A)$, the greater the CPU time needed to solve DP by MBR.

It follows that each function $h_{MBR}(DP)$ that is able to estimate the cardinality of $COVS_{DP}(OBS^A)$ for each diagnostic problem DP can be used to estimate the computational effort required to solve DP by MBR.

In order to be effectively used in the flexible architecture described in Section 7, the computation of such a function should require little time. However, the oracle which the flexible architecture is based on does not need a precise estimation of the CPU time and the heuristic function needs only to be accurate enough to allow a good classification of each diagnostic

problem into one of the three classes: the classes of problems whose resolution by MBR is *easy*, *medium* or *hard*. This fact gives us enough space for a finding a good trade-off between the accuracy of the heuristic function in estimating the cardinality of the set $COVS_{DP}(OBS^A)$ and the time required to compute it. In particular, the oracle in the flexible architecture makes use of the following heuristic function:

Definition B.1 *Given a diagnostic problem* $DP = \langle BM, HYP, CXT, \langle OBS^A, \Psi^- \rangle \rangle$,
$h_{MBR}(DP) = \prod_{m(a) \in OBS^A} card(\overline{MIN\_COVS}_{DP}(\{m(a)\}, \Psi^-))$, *where*
$card(\overline{MIN\_COVS}_{DP}(\{m(a)\}, \Psi^-))$ *denotes the cardinality of the set* $\overline{MIN\_COVS}_{DP}(\{m(a)\}, \Psi^-)$.

It is easy to see that such a heuristic function over-estimates the cardinality of the set $COVS_{DP}(OBS^A)$. Indeed, $h_{MBR}(DP)$ actually estimates the cardinality of the set $S_{DP}(OBS^A)$ computed in the sub-step 1.2 which is a superset of $COVS_{DP}(OBS^A)$. It is worth noting that the above-defined $h_{MBR}(DP)$ function considers the sets $\{\overline{MIN\_COVS}_{DP}(\{m(a)\}, \Psi^-) / \ m(a) \in OBS^A)$ and not the sets
$\{MIN\_COVS_{DP}(\{m(a)\}) / \ m(a) \in OBS^A)\}$, since it takes into account the necessary conditions used by the MODEL-BASED REASONER to focus the search.

The time for computing the heuristic function $h_{MBR}(DP)$ is very limited since the set of minimal coverings $MIN\_COVS(\{m(a)\})$ for each abnormal value a and each observable parameter m depends only on BM (thus we can omit the subscript DP, since these sets of coverings hold for every diagnostic problem relevant to BM) and it can be computed off-line; moreover, for every $Cov \in MIN\_COVS(\{m(a)\})$ the set $Nec(Cov)$ of the necessary conditions associated with Cov are pre-compiled. Therefore, the computation of $h_{MBR}(DP)$ requires just to count, for each $MIN\_COVS(\{m(a)\})$ $(m(a) \in OBS^A)$, how many $Cov \in MIN\_COVS(\{m(a)\})$ are such that $Nec(Cov) \cap \Psi^- = \emptyset$ and to compute the product of these numbers. These operations are all inexpensive and in fact, the experimental results show that the evaluation of $h_{MBR}(DP)$ is performed in at most 5 msec. which is an almost negligible fraction of time with respect to the whole time needed for solving each diagnostic problem.

Despite the simplicity of the above-defined heuristic function, their values still have a satisfactory correlation with the CPU times required by the MODEL-BASED REASONER to solve the problems (see [28]); moreover, the oracle defined in Section 7.1 (that makes use of this function in order to predict the class for each diagnostic problem) is quite effective (see Table 8).

B.1 Learning Oracle Thresholds

The function $Oracle(DP)$ that predicts the class for each diagnostic problem DP makes use of two thresholds S_1 and S_2 (Section 7.1): if $h_{MBR}(DP) \leq S_1$ or $h_{MBR}(DP) \geq S_2$ the problem DP is classified as *easy* or *hard*, respectively; otherwise DP is classified as *medium*. The two thresholds are automatically learnt on the basis of a test set TS. TS is a set of pairs $\langle Class(DP), h_{MBR}(DP) \rangle$, where $Class(DP)$ is the actual class which the diagnostic problem DP belongs to and $h_{MBR}(DP)$ is the value for DP of the heuristic function defined in the previous section.

The learning algorithm considers the following error function [16]:
$error(x, y) = \sum_{I \in \{E,M,H\}} \frac{card(miscl_I(x,y))}{card(I)}$

[16] We assume that in the training set there is at least one problem for each class; i.e. for each $I \in \{E, M, H\}$ $card(I) > 0$.

where E indicates the set of pairs $\langle easy, h_{MBR}(DP)\rangle \in TS$ (M and H stay for *medium* and *hard*, respectively) and $miscl_I(x, y)$ is the set of problems in I that are misclassified by the function $Oracle(DP)$ using the thresholds $S1 = x$ and $S2 = y$. The learning mechanism searches the following domain Dom for a pair $\langle S1, S2\rangle$ of thresholds that minimizes the error function:

$Dom = \{\langle x, y\rangle / \langle x, y\rangle \in D_{S1} \times D_{S2} \wedge x < y\}$,

where

$D_{S1} = \{h_{MBR}(DP) / \langle easy, h_{MBR}(DP)\rangle \in TS\}$

and

$D_{S2} = \{h_{MBR}(DP) / \langle hard, h_{MBR}(DP)\rangle \in TS\}$.

If $Dom = \emptyset$, the thresholds cannot be learnt. In practice, given the good correlation between the CPU time and the value of the heuristic function $h_{MBR}(DP)$ each test set containing a representative sample of cases prevents this situation to occur.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations and system approaches. *AI Communications*, 7(1):39–59, 1994.

[2] D. Aha. The omnipresence of CBR in science and application. Technical Report AIC-98-002, Navy Center for Applied Research in AI, 1998.

[3] D. Aha and J. Daniels (eds.). *Proc. AAAI Workshop on CBR Integrations.* AAAI Press, 1998.

[4] A. An, N. Cercone, and C. Chan. Integrating rule induction and case-based reasoning to enhance problem solving. In *Proc. 2nd International Conference on Case-Based Reasoning, LNAI 1266*, pages 499–508, Providence, RI, 1997. Springer.

[5] T.C. Au, H. Munoz-Avila, and D.S. Nau. On the complexity of plan adaptation by derivational analogy in a universal classical planning framework. In *Lecture Notes in Artificial Intelligence 2416*, pages 13–27. Springer Verlag, 2002.

[6] J.L. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity I.* Springer Verlag, 1988.

[7] B. Bartsch-Spoerl. Towards the integration of case-based, schema-based and model-based reasoning for supporting complex design tasks. In *LNAI 1010*, pages 145–156. Springer Verlag, 1995.

[8] R. Bellazzi, S. Montani, L. Portinale, and A. Riva. Integrating rule-based and case-based decision making diabetic patient management. In *Proc. 3th International Conference on Case-Based Reasoning, LNAI 1650*, pages 386–400. Springer Verlag, 1999.

[9] I. Bichindaritz, E. Kansu, and K.M. Sullivan. Case-based reasoning in CARE-PARTNER: gathering evidence for evidence-based medical practice. In Springer Varlag, editor, *Proc. 4th EWCBR, LNAI 1488*, pages 334–345, 1998.

[10] L.K. Branting and B.W. Porter. Rules and precedents as complementary warrants. In *Proc. 9th National Conference on Artificial Intelligence (AAAI 91)*, Anaheim, 1991.

[11] J.S. Breese and D. Heckermann. Decision-theoretic case-based reasoning. In *Proc. 5th International Workshop on AI and Statistics*, Fort Lauderdale, FL, 1995.

[12] T. Bylander, D. Allemang, M. Tanner, and J. Josephson. The computational complexity of abduction. *Artificial Intelligence*, 49(1-3):25–60, 1991.

[13] L. Console, L. Portinale, and D. Theseider Dupré. Using compiled knowledge to guide and focus abductive diagnosis. *IEEE Transactions on Knowledge and Data Engineering*, 8(5):690–706, 1996.

[14] L. Console and P. Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7(3):133–141, 1991.

[15] L. Console and P. Torasso. An approach to the compilation of operational knowledge from causal models. *IEEE Trans. on Systems, Man and Cybernetics*, 22(4):772–789, 1992.

[16] W.F. Dowling and J.H. Gallier. Linear time algorithms for testing the satisfiability of propositional Horn clauses. *Journal of Logic Programming*, 1:267–284, 1984.

[17] D.B. Leake (ed.). *Case Based Reasoning: Experiences, Lessons and Future Directions.* AAAI Press, 1996.

[18] A.G. Francis and A. Ram. The utility problem in case-based reasoning. Technical Report ER-93-08, Georgia Tech, 1993.

[19] E. Freuder. *AAAI Spring Symposium on Multi-modal Reasoning.* AAAI Press, 1998.

[20] A. Goel. Integration of case-based reasoning and model-based reasoning for adaptive design problem solving. Technical report, (PhD Diss.), Ohio Univ., 1989.

[21] A. Goel, K. Ali, M. Donnellan, A. deSilva Garza, and T. Callantine. Multystrategy adaptive path planning. *IEEE Expert*, 9(6):57–65, 1994.

[22] K.J. Hammond, T. Converse, M. Marks, and C. Seifert. Opportunism and learning. *Machine Learning*, 10(3), 1993.

[23] J.L. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann, 1993.

[24] P. Koton. Using experience in learning and problem solving. Technical report, MIT/LCS/TR-441, 1989.

[25] D.B. Leake and A. Kinley. Combining reasoning modes, levels and styles through internal cbr. In *Proc. AAAI Spring Symposium on Multi-modal Reasoning*, Stanford, 1998. AAAI Press.

[26] D.B. Leake, B. Smith, D.C. Wilson, and Q. Yang (eds.). *Computational Intelligence: special issue on Maintaining Case-Based Reasoning Systems.* volume 17, nuber 2, 2001.

[27] P.J.F. Lucas. Analysis of notion of diagnosis. *Artificial Intelligence*, 105:295–343, 1998.

[28] D. Magro and P. Torasso. Performance issues in unimodal and multimodal reasoning approaches to diagnosis. In *Proc. 11th Intl. Work. on Principles of Diagnosis, DX'00*, pages 109–116, Morelia, Mexico, 2000.

[29] C. Marling, M. Sqalli, E. Rissland, H. Munoz-Avila, and D. Aha. Case-Based Reasoning integrations. *AI Magazine*, 23(1):69–86, 2002.

[30] S. Minton. Qualitative results concerning the utility of EBL. *Artificial Intelligence*, 42:363–391, 1990.

[31] B. Nebel and J. Koehler. Plan reuse versus plan generation: a theoretical and empirical analysis. *Artificial Intelligence*, 76:427–454, 1995.

[32] L. Portinale and P. Torasso. ADAPtER: an integrated diagnostic system combining case-based and abductive reasoning. In *Proc. 1st ICCBR, LNAI 1010*, pages 277–288. Springer Verlag, 1995.

[33] L. Portinale and P. Torasso. On the usefulness of re-using diagnostic solutions. In *Proc. 12th European Conf. on AI - ECAI 96*, pages 137–141, Budapest, 1996.

[34] L. Portinale and P. Torasso. Performance issues in ADAPtER a combined CBR-MBR diagnostic architecture. In *Proc. AAAI Spring Sympos. on Multi-Modal Reasoning*. AAAI Press, Stanford, 1998.

[35] L. Portinale and P. Torasso. Case base maintenance in a multimodal reasoning system. *Computational Intelligence*, 17(2):263–279, 2001.

[36] L. Portinale, P. Torasso, and D. Magro. Selecting most adaptable diagnostic solutions through Pivoting-Based Retrieval. In *Proc. 2nd ICCBR, LNAI 1266*, pages 393–402. Springer Verlag, 1997.

[37] L. Portinale, P. Torasso, C. Ortalda, and A. Giardino. Using case-based reasoning to focus model-based diagnostic problem solving. In *LNAI 837*, pages 325–337. Springer Verlag, 1994.

[38] L. Portinale, P. Torasso, and P. Tavano. Dynamic case memory management. In *Proc. ECAI 98*, pages 73–78, Brighton, 1998.

[39] E.L. Rissland and D.B. Skalak. Combining case-based and rule-based reasoning: a heuristic approach. In *Proc. 11th IJCAI*, pages 524–530, Detroit, 1989.

[40] B. Selman and H. Levesque. Abductive and default reasoning: A computational core. In *Proc. AAAI 90*, pages 343–348, Boston, 1990.

[41] B. Smyth and P. Cunningham. The utility problem analysed: a case-based reasoning perspective. In *LNAI 1168*, pages 392–399. Springer Verlag, 1996.

[42] B. Smyth and M.T. Keane. Design a la Deja' Vu: reducing the adaptation overhead. In D.B. Leake, editor, *Case Based Reasoning: Experiences, Lessons and Future Directions*. AAAI Press, 1996.

[43] M.H. Sqalli, L. Purvis, and E.C. Freuder. Survey of applications integrating constraint satisfaction and case-based reasoning. In *Proc. PACLP99, First Int. Conference and Exhibition on the Practical Application of Constraint Technologies and Logic Programming*, London, 1999.

[44] R. Straatman and P. Beys. A performance model for knowledge-based systems. In *Proc. EUROVAV-95, European Symposium on the Validation and Verification of Knowledge-Based Systems*, pages 253–283, Chambery, 1995.

[45] J. Surma and K. Vanhoff. Integrating rules and cases for the classification task. In *Proc. 1st ICCBR, LNAI 1010*, pages 325–334. Springer Verlag, 1995.

[46] P. Torasso, L. Portinale, L. Console, and M. Casassa Mont. Approximate reasoning in a system combining prototypical knowledge with case-based reasoning. In L.A. Zadeh and J. Kacprzyk, editors, *Fuzzy Logic for the Management of Uncertainty*. John Wiley & Sons, 1992.

[47] F. van Harmelen. A model of costs and benefits of meta-level computation. In *Proc. Fourth Workshop on Meta-programming in Logic (META '94), LNCS 883*, pages 248–261. Springer Verlag, 1994.

[48] M. van Someren, J. Surma, and P. Torasso. A utility-based approach to learning in a mixed case-based and model-based reasoning architecture. In *Proc. 2nd ICCBR, LNAI 1266*, pages 477–488. Springer Verlag, 1997.

[49] M. vanSomeren, P. Reimann, H.P.A. Boshuizen, and T. deJong (eds.). *Learning with Multiple Representation.* Elsevier, 1998.

[50] M. Veloso. *Planning and learning by analogical reasoning.* LNAI 886, Springer Verlag, 1994.

[51] D.C. Wilson and D.B. Leake. Maintaining case-based reasoners: dimensions and directions. *Computational Intelligence*, 17(2):196–213, 2001.