Case-based learning of plans and goal states in medical diagnosis

Beatriz López[*], Enric Plaza
Institut d'Investigació en Intel·ligència Artificial, CSIC
Campus Universitat Autònoma de Barcelona,
08193 Bellaterra, Spain
Tel.: + 34 3 580 95 70
e-mail: blopez@etse.urv.es,enric@iiia.csic.es

Abstract

We introduce a case-based system, BOLERO, that learns both plans and goal states, with the aim of improving the performance of a rule-based system by adapting the rule-based system behavior to the most recent information available about a patient. On the one hand, BOLERO gets knowledge from cases in the form of plans that are represented as sequences of decision steps. The advantages of this representation include: 1) retrieval and adaptation of parts of plans (steps) appropriate to the current problem state; 2) generation of new plans not previously available in memory; and 3) learning from experience, either from successful or failed plans. On the other hand, goal states are sets of diagnosis likelihoods and are not known beforehand. For this reason BOLERO uses solutions of past cases to recognize a state as a goal state of a new planning problem. BOLERO and a rule-based system are integrated into a meta-level architecture in which we emphasize the collaboration of both systems in solving problems. The rule-based system executes the plans generated by BOLERO. As a consequence of the execution of plans, the rule-based system furnishes BOLERO with new information with which BOLERO can generate a new plan to adapt the reasoning process of the rule-based system into correspondence with the recent available data. All the methods have been designed to be useful for medical diagnosis and have been tested in the domain of diagnosing pneumonia.

Keywords: case-based reasoning, adaptive planning, medical diagnosis, goal state learning, learning from experience.

[*]Current address: Departament d'Enginyeria Informàtica, Escola Tècnica Superior d'Enginyeria, Universitat Rovira i Virgili, Carretera de Salou, s/n, 43006 Tarragona, Spain.

1 Introduction

Advances in AI technology allows the development of complex, real world applications. In knowledge-based systems (KBS) of this sort, planning plays an important role insofar as plans can make problem solving more efficient, controlling the resources needed to solve a problem and adapting the problem solving process to changes in the environment (Tate, 1990).

An adaptation to environmental changes is especially important in KBS's in medical domains where the data available is usually incomplete and uncertain. The physician copes with incompleteness and uncertainty by dynamically generating and executing plans . For example, when a patient arrives at the hospital, the information usually available consists of the name of the patient, age, social security number, etc. The physician then performs an exploratory examination to determine the condition of the patient. To diagnose an illness, each physician follows a set of questions, for example, the pathological record of interest of the patient, (smoker, alcoholic, diabetic, ...), that leads him to generate hypotheses about the patient's illness. Having a set of hypotheses, the physician plans a set of clinical tests and physical examinations in order to collect data bringing forward some evidence confirming or discarding the hypotheses. At the same time, the new data available can suggest new and more likely hypotheses to the physician. Then, he plans a new set of tests and examinations in order to confirm or disprove the new set of hypotheses. This process continues until either a set of hypotheses has been confirmed with a certain degree of evidence, or the physician is unable to determine the patient's illness. Thus, an important issue when planning the task of a medical diagnosis KBS is the ability to adapt the system behavior to the most recent data available by re-planning.

Another interesting point in developing a planning system in medical domains is the fact that there is no goal state defined as part of the given data of the problem to be solved. Whereas a planning problem is typically defined by a initial state, a set of operators and a specification of goal states, in medical diagnosis we do not have these final state specification as input to the planner [1]. A physician plans the sequence of decision steps that leads him to conclude the diagnosis of a patient without knowing in advance which this diagnosis will be. He explores different alternatives and finally, he founds a diagnosis with a certain degree of certainty. We say that in medical diagnosis we have an *open-goal planning* problem. For this reason when planning in medical diagnosis we need a method that learns to recognize when a satisfactory solution to a problem has been achieved.

The most challenging question in developing a planning system is that it is preferable that the system learns the knowledge needed to plan rather than requiring a knowledge engineer to acquire and code this knowledge. Plans can make problem solving more efficient but we observed in earlier experiments at our Institute that the process of assuring the proper sequence of actions during problem solving strongly depends on the skills of the knowledge engineer to "tune" the system. An empirical study made at Carnegie Mellon University (Carbonell, 1991) points out that the knowledge engineer spends about 30% to 60% of the time performing such "tuning". It seems convenient to provide methods that acquire the strategic knowledge required to plan, eliminating the hand-made tuning phase.

Taking into account the previous considerations, we focus our research on learning plans and learning goal states in medical diagnosis. As a result of our work, we present on this paper four main contributions. First, we have developed a case-based planning method that eases the knowledge acquisition phase of complex, real world applications. With such a method it is possible to learn plans from cases provided either by a teacher or by the system's own experience. Advantages of case-based planning are saving engineer's time in knowledge acquisition and avoiding the detection and correction of interactions among different parts of knowledge (i.e. the "tuning" work).

The second contribution is that BOLERO, the case-based planner we built, is able to retrieve plans from memory according to the information available anytime. In order to plan a task (eg.

[1]Moreover we show later in this paper how we do not deal with operators the effects of which are known before its execution, but with actions the effect of which are not known beforehand.

diagnosing of a patient), BOLERO re-uses plans and adapts the system behavior to changes in the environment by re-planning. The approach followed to represent cases has been relevant to reuse *parts* of past plans to solve a problem.

The third contribution is a meta-level architecture in which we integrate the case-based planner with an existing rule-based system (RBS diagnostic system). In such architecture BOLERO learns how to plan the task the RBS performs in order to improve the RBS performance. The solution of a problem is then achieved by interleaving planning (by the case-based planner) and execution of plans (by the RBS), in such a way that the reasoning process is adapted to changes in the environment. Uncertainty and incompleteness typical in medical diagnosis is handled in a traditional approach to the rule-based system with fuzzy methods (Agustí et al., 1992) or certainty factors (Pearl, 1988) whereas re-planning offers a new approach to fit the diagnostic procedure to the last relevant information available.

Finally, we have developed a case-based method to learn and recognize goal states. In medical diagnosis a goal state is defined as a sufficiently accurate set of diagnoses. This case-based method runs concurrently with the case-based planner, and decides upon problem solving termination. The key issue of the method relies on the measures defined to compare a past goal state with the current problem solving state in order to acknowledge it as a goal state. In addition to learn plans, BOLERO then learns to recognize appropriate solutions to problems.

1.1 Overview of the rest of the article

We continue this paper by describing the key issue in which all of our work is founded, that is, the representation of plans as cases. Then, we explain the meta-level architecture that constitutes the framework in which the case-based system and the knowledge-based system are integrated. After that, we proceed by describing BOLERO first as the system that learns and generates plans, and second as the system that learns and recognizes goal states. Then we give the results obtained when we applied BOLERO to a medical domain. And finally, we discuss the relations of which are the relations of our system to other research, the conclusions of our work, and the open problems and ideas that we want to continue from now on.

2 BOLERO's cases

Cases are the elemental entities in which case-based reasoners work. So when designing a case-based system we must pay special attention to case representation. In our domain, problems are represented by the pathological record of a patient, in which data gathered during the diagnosis are registered, as well as the analysis and any kind of probes performed to obtain them. In the record of a patient we can also find the diagnosis, if achieved, and the suggested therapy.

BOLERO cases come from two sources. On one hand, cases solved by the system can be added to its memory, learning by its own experience. On the other hand, when the system has no cases, a teacher provides a set of initial cases. Those cases are used to train the system in problem solving. The quality and complexity of the training cases, as well as the order in which they are proportioned to the system, are really relevant, as shown in (Cornuejols, 1993).

In the introduction, we have shown how a physician diagnoses a patient by following a sequence of decision steps in which the physician successively adapts plans. Consistently, if we want to capture in a case the adaptive behavior of the physician when diagnosing a patient, we must reflect the different steps in which the physician plans new tests and examinations. In each step of the diagnostic process we need to distinguish the data or known facts, the situations or set of facts in which the physician changes his hypotheses, and the actions (set of tests and examinations) which confirm or discard them. In a case, it is important to distinguish the diagnosis of the patient or solution of the problem from the plan followed to achieve it. All of these points are explained in the following, and we end this section by giving the case representation approach followed in BOLERO, and how we have acquired the cases from the expert.

2.1 Facts

A fact can be described by an attribute-value pair, as for example *(pathological-record, drug-addict)*. However this description is poor, mainly for two reasons.

First of all, the physician distinguishes relevant facts, as for example, *(hemoculture, yes)*, from less-relevant ones, as for example *(pathological-record, {smoker, drug addict})*. The physician relies more on the fact *hemoculture* than on the fact *pathological-record* because the hemoculture is a specific test, the results of which provide precise information about the patient's illness. However, hemocultures are only performed 3,79% of the times, meanwhile more than 86% of the times the pathological history of the patient is known. Specific data should be the most relevant, but they are typically known in few occasions. So it is necessary in the representation of a fact to take into account the relevance and for our purposes, especially the relevance from the viewpoint of planning.

Secondly, attributes of facts are related to each other. Relations can be of different types: hierarchical, precedence, etc. For example, *associated clinical data* (i.e. skin and circulatory) and *dispnea* belong to the *semiology* category. Relations among attributes help the physician to know the relevance of facts. For example, *semiology* data is not always reliable because its interpretation is quite subjective, and so it has a lesser importance. Therefore, *associated clinical data* and *dispnea* are also less important. We take into account relations of facts by expressing them explicitly in a domain network, as the one shown in figure 1. Beyond facilitating the strategic knowledge acquisition of facts, the network is helpful to perform abstraction when learning from cases.

It is interesting to note that relations are useful both for learning and inference purposes. For example, inference can use relations for avoiding asking about a specific fact if some previous, more general one, is still unknown. So if we try to learn plans for an existing rule-based system, we can assume that we can use the existing domain network.

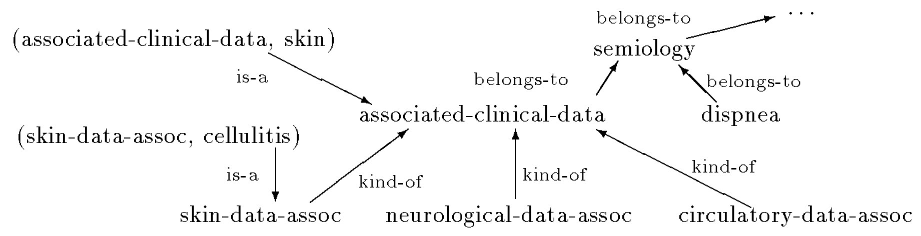


Figure 1: Partial view of the domain knowledge network where different relations among attributes are defined (*belongs-to*, *kind-of*, etc.) Facts are connected with a *is-a* link to the attribute.

2.2 Situations and actions

A situation is a set of facts. Given a situation, we can have a set of hypotheses. Then what we need is an action to confirm or discard the hypotheses, that is, some test or examination. In each situation the physician can plan a sequence of actions $[a_1, \ldots, a_n]$ or plan, p. When executing the actions, new evidence about the patient's illness, and so, about the hypotheses, can be found. Situation changes occur. As a consequence, new, different actions can be planned.

Let us suppose that in a given situation the physician is trying to discriminate the type of illness the patient suffers: either bacterial or atypical pneumonia. This is called the *bact-atyp* action and *bacterial-pneumonia* and *atypical-pneumonia* are facts that, when instantiated, constitute the results of the actions. To instantiate them, the physician plans a gathering of information about

Patient b06v1:	Patient g01v1
(a) **Initial data**	
(community-acquired, *yes*)	(community-acquired,*yes*)
(sex, *male*)	(sex, *male*)
(establishment, *sudden*)	(establishment, *sudden*)
(b) **Data gethered during execution**	
(state-of-patient, *mod-serious*)	(state-of-patient, *slight*)
(age, *85*)	(age, *27*)
(asylum, *unkown*)	
(X-rays, *ill-defined-bound, confluent-images*)	(X-rays, *ill-defined-bound, interstitial-pattern*)
(bacterial-pneumonia, *very-possible*)	(atypical-pneumonia, *very-possible*)
(c) **Final plan**	
[gather-information	[gather-information
bact-atyp	bact-atyp
pneumoccoccus	mycoplasma
enterobacteriaceae	virus
legionella	chlamydia]
anaerobis	
tuberculosis	
staphiloccoccus	
s-pyogenes	
hemophylus	
meningoccoccus	
brahnamella]	
(d) **Diagnostics**	
(pneumonia-pneumoccoccus, *possible*)	(pneumonia-mycoplasma, *mod-possible*)
(pneumonia-hemophylus, *slightly-possible*)	(viral-pneumonia, *quite-possible*)
	(pneumonia-chlamydia, *mod-possible*)

Figure 2: Data of patients b06v1 and g01v1. (a) Initial state. (b) Data gathered during execution. (c) Solution from the planning point of view (or final plan). (d) Solution from the domain point of view or diagnosis.

the patient by filling in his/her pathological record, doing an x-ray photograph of the thorax, and other tests and examinations.

It is interesting to note here, and to keep in mind in reading the paper, that planning in medical diagnosis deals with actions, while most planning systems manipulate operators. The results of the operators are known. The results of actions are not, since it is the environment that responds specifying the results of an action[2].

2.3 Diagnostic procedures and solved problems

In medical diagnosis, a problem consists of finding the agent that causes a patient's illness, or diagnosing the patient's illness itself. Situations and actions are elementary steps in the overall diagnostic process or plan that leads to determine the patient's illness. In general, all physicians have an standard diagnostic procedure from which to start the diagnosis, and that is successively adapted to the current patient according to the particular data being gathered. For example, we show in figure 2 the data of two patients that arrive at the hospital. Initially, (fig. 2a) the physician knows the same data for both, and then executes the same initial plan: *[gather-information, bact-atyp]*. However, those two patients have finally different diagnoses (fig. 2d), and the diagnostic procedure followed by the physician to diagnose them has been different (fig. 2c). This is a consequence of the adaptive behavior followed to solve problems. The data gathered when applying the same actions, *gather-information* and *bact-atyp*, is different in each patient. The physician after knowing whether it is a bacterial or an atypical pneumonia (case *b06v1* and case *g01v1* of figure 2 respectively) plans different actions to diagnose each patient.

From the point of view of the domain knowledge, we can say that the solution is that the patient suffers illness A or B, with a certain degree of certainty. However, from the point of view of planning, the solution to the problem is the sequence of actions (i.e. tests and examinations) performed from the beginning to the end of the diagnosis. That is, the solution from the planning

[2]An action is a process to be performed in the real world

point of view is not the initial plan (since we have seen that it is the same for both patients of figure 2), but the final sequence of actions effectively executed. For example, the solution from the planning point of view of patient *g01v1* of figure 2 is the plan: *[gather-information, bact-atyp, mycoplasma, virus, chlamydia]*. And the solution from the domain point of view is the set of possible diagnoses { *(pneumonia-by-mycoplasma almost-sure), (viral-pneumonia possible), (pneumonia-by-chlamydia slightly-possible)* }.

2.4 Case representation

To capture the adaptive behavior shown by the physician when diagnosing a patient, some recent works, as for example (López & Plaza, 1989),(Redmond, 1990) and (Veloso, 1992) explore the representation of cases based on *snippets* (pieces of cases) and episodes. The approach taken in BOLERO follows the same line: a case is represented as a sequence of decision steps or *episodes*:

$$C^i = [\delta_0^i, \delta_1^i, \ldots, \delta_{n_i}^i]$$

Each episode $\delta_j^i = \langle s_j^i, p_j^i \rangle$ is a pair formed by a situation s_j^i and a plan p_j^i. The situation s_j^i is the representation of the known facts of the patient i in the episode j. The plan p_j^i is not the plan that the physician executes from this moment on, but something different. The plan p_j^i is composed of the set of actions performed up to the moment $a_0^i, \ldots, a_{j-1}^i$, plus the action a_j^i from which we proceed with the diagnosis. This representation allows us to have in the last episode, $\delta_{n_i}^i$, the final plan, $p_{n_i}^i$ or plan that, when executed from the initial situation s_0^i, leads us to know the patient's illness. Any other plan p_j^i ($j \neq n_i$) of other episodes is considered as a partial plan. The final plan $p_{n_i}^i$ is the solution of the case from the planning point of view, s_0^i is the initial state and $s_{n_i}^i$ is the final state or goal state. The goal state contains the solution (set of facts) from the domain point of view.

If a plan is in an episode δ_j^i that precedes another episode δ_k^i, $\delta_j^i \prec \delta_k^i$, then the plan p_k^i is longer that the plan p_j^i in such a way that

$$p_k^i = [a_0, \ldots, a_{j-1}, a_j, a_{j+1}, \ldots, a_k]$$

$$p_j^i = [a_0, \ldots, a_{j-1}, a_j]$$

Particularly, if $\delta_j^i \prec \delta_{j+1}^i$, then the plan p_{j+1}^i and the plan p_j^i only differ in that the last action of p_{j+1}^i is not in p_j^i. Analogously, we say that a situation s_j^i precedes another one s_k^i, $s_j^i \prec s_k^i$, if we can say that the episode $\delta_j^i \prec \delta_k^i$. Situations are assumed to be monotonic: if $s_j^i \prec s_k^i$ then any fact of s_j^i is also in the situation s_k^i.

Any fact in a situation of a case is represented by an attribute and a value, $\langle at, v \rangle$. The different relations existing among attributes have been acquired from the domain expert and are represented in a domain network (see figure 1). Facts can be boolean (*(cough, yes)*), set-valued (*(pathological-record, { smoker, drug-addict })*), numerical (*(age, 51)*), or fuzzy (*(bacterial-pneumonia, possible)*). In particular, fuzzy facts are important since they are strictly related to the diagnosis finally achieved for a given patient. The approach taken in BOLERO follows from the shell MILORD (Agustí et al., 1992), where fuzzy values for medical diagnosis are the following set of linguistic labels: { *impossible, almost-impossible, slightly-possible, possible, quite-possible, very-possible, almost-sure, sure* }. Two fuzzy facts can be compared according to the position of their certainty value inside this set of linguistic labels. For example, the value of the fact *(atypical-pneumonia, possible)* is less than the value of the fact *(atypical-pneumonia, sure)*. Analogously, we can say that the distance between the values of those facts, $d(possible, sure)$, is 4. These properties of fuzzy facts have been exploited when building similarities functions to retrieve cases from memory, as explained later.

In the examples of figure 3, the case C^{g01v1} is composed of six episodes: $C^{g01v1} = \{\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6\}$[3]. The episode δ_2 is formed by the situation s_2 and plan p_2, where: $s_2 =$

[3] Note that although this representation seems redundant, it is not implemented this way but by using inheritance techniques.

{ *(severity, light), (age, 85), (asylum, unknown), (RX-data, {bad-def-bound, confluent })* }and p_2 = *[gather-information, bact-atyp]*. The fact *(contact-with-pneumonia, no)* is boolean, *(severity, light)* set-valued, *(age, 85)* numeric, and *(atypical-pneumonia, very-possible)* fuzzy.

Episode:	Facts and plans:	
	(a) b06v1	(b) g01v1
δ_1:	{ (community-acquired, *yes*) (sex, *man*) (onset-form, *sudden*) } [gather-information]	{ (community-acquired, *yes*) (sex, *man*) (onset-form, *sudden*)} [gather-information]
δ_2:	{ (severity, *somewhat-serious*) (age, *85*) (asylum, *unknown*) (RX-data, *bad-def-bound, confluents*) } [gather-information, bact-atyp]	{ (severity, *light*) (age, *27*) (RX-data, *bad-def-bound, interstitial*) } [gather-information, bact-atyp]
δ_3:	{ (headache, *unknown*) (bacterial-pneumonia, *very-possible*) } [gather-information, bact-atyp, pneumococ]	{ (headache, *unknown*) (atypical-pneumonia, *very-possible*) } [gather-information, bact-atyp, mycoplasma]
δ_4:	{ (pneumococcus, *possible*) } [gather-information, bact-atyp, pneumococ, enterobacteriac]	{ (contact-with-pneumonia, *no*) (mycoplasma-pneumonia, *slightly-possible*) } [gather-information, bact-atyp, mycoplasma, virus]
δ_5:	{ (jaundice, *unknown*) (enterobacteriaceae, *almost-impossible*) } [gather-information, bact-atyp, pneumococ, enterobacteriac, legionella]	{ (perihilars, *unknown*) (viral-pneumonia, *quite-possible*)} [gather-information, bact-atyp, mycoplasma, virus, chlamydiae]
δ_6:	. . . [gather-information, bact-atyp, pneumococ, enterobacteriac, legionella, . . .]	{ (chlamydiae-spp, *slightly-possible*) } [gather-information, bact-atyp, mycoplasma, virus, chlamydiae, □]

Figure 3: Representing cases b06v1 and g01v1 by episodes. Situations are enclosed in brackets, facts in round brackets, and plans in square brackets.

2.5 Acquiring cases from experts

Automatic generation of cases is not an easy task when dealing with real world applications, due to the presence of contradictory data, exclusive values, etc. For this reason we assume that there is a human expert that can provide cases.

Cases can be easily acquired from the expert by using the representation provided in the previous section and based on the hospital records of patients. We reconstruct the decision steps that determine the physician adaptive behavior by interviewing him in mock up sessions. First, we show part of the patient data to the physician and we ask for his initial hypotheses as well as the actions that he would do to confirm them. Then we show to the expert information about the patient resulting from those actions, and then we ask him the next set of hypotheses. We follow asking him his next actions with which we build the next decision step. This process continues until a solution is achieved.

3 The meta-level architecture: Interleaving plan elaboration and plan execution

The integration of the case-based system BOLERO and a rule-based system (RBS) is realized by means of a meta-level architecture. For the sake of simplicity, we call BOLERO-RBS the resulting system. In such architecture, BOLERO plays the role of the meta-level and the RBS plays the role of the object-level (figure 4). The RBS has knowledge about a specific domain in order to solve a problem while BOLERO has planning knowledge particular to that domain. We continue by describing each level of the architecture and then we show the BOLERO-RBS behavior when solving problems.

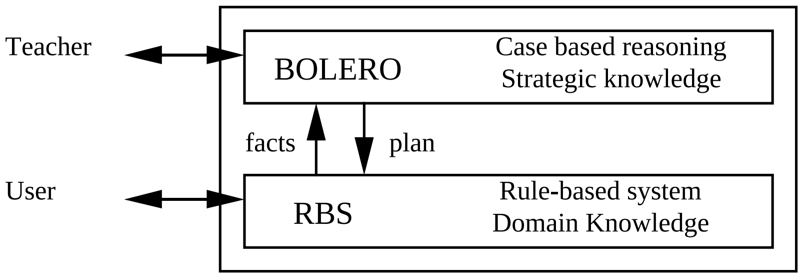


Figure 4: The BOLERO-RBS system.

3.1 The object-level

As we have already mentioned, the object-level is a rule-based system and, as any rule-based system, it has an interpreter and a knowledge base. The knowledge base has rules (R) and facts. Particularly we distinguish three kind of facts: (a) input facts (F_i) or facts that can be gathered from the user; (b) deducible facts (F_d) or facts that can be concluded from rules, and (c) output facts or goals (F_g), that is, facts that constitute the solutions to problems and that can be also concluded from rules ($F_g \in F_d$). In figure 5 examples of rules, facts and goals of the rule-based system used in our research are shown. The RBS is able to work independently of BOLERO-RBS. That is, the RBS has a plan by default that consists of the sequence of all possible goals. The interpreter of the RBS solves a problem by using the rules that allow it to deduce all the goals specified in the default plan.

R01010b:	If *age* $\in$ [60,64]
	Then conclude *old-person* is *certain*
R04009:	If *old-person* $\wedge$ *bacterial-pneumonia* is *possible*
	Then conclude *pneumonia-pneumococcus* is *possible*

Figure 5: Rules of the PNEUMON-IA system. *Age* is an input fact, *old-person* is a deducible fact, and *bacterial-pneumonia* and *pneumonia-pneumoccoccus* are goal facts.

3.2 The meta-level

BOLERO constitutes the meta-level of the BOLERO-RBS system. BOLERO as a case-based planner learns strategic knowledge in the form of plans from cases and then it builds new plans to solve new problems. To be able to plan the goals of the object-level, BOLERO needs a partial model of the RBS. This model allows BOLERO to reason about the object-level problem solving state, and act consistently. The partial model that BOLERO knows of the RBS is quite simple: BOLERO knows the input (F_i) and output facts (F_g) of the RBS. A fact $f_i \in F_i$ of the object-level is represented in BOLERO as a fact $\lceil f_i \rceil$, where $\lceil f_i \rceil$ is the quotation of f_i, that is, the representation of f_i in the language of BOLERO. Analogously, a fact $f_g \in F_g$ in the RBS is represented as $\lceil f_g \rceil$ in BOLERO. In addition to input and output facts, BOLERO can also access relations among attributes that establish a domain network (see section 2).

BOLERO also knows the set of actions that can be performed in the RBS. Each action a in BOLERO is interpreted by the RBS as pursuing a goal $f \in F_g$. A plan p in BOLERO is formed by the sequence of actions $a_1, a_2, \ldots, a_n$ is equivalent to a plan $\lfloor p \rfloor$ in the RBS formed by the goals $f_{g_1}, f_{g_2}, \ldots f_{g_n}$[4]. The instantiate of the goal fact f_{g_i} by RBS is interpreted as the execution of action a_i in BOLERO.

[4]We use Kleene's $\lceil$quotes$\rceil$ and $\lfloor$unquotes$\rfloor$ for the metalevel lift-up and lift-down.

3.3 Integration: Adaptive planning

BOLERO-RBS has a set of reflection rules, as any meta-level system, that assures the causal connection between both levels, and that determine when the control of the overall system changes from one level to the other (Maes, 1988). Reflection rules are the following:

$$R_{u_i}\colon \frac{f_i}{\lceil f_i \rceil} \qquad R_{u_g}\colon \frac{f_g}{\lceil f_g \rceil} \qquad R_d\colon \frac{p}{\lfloor p \rfloor}$$

The upwards reflection rules, R_{u_i} and R_{u_g}, carry up the information present at the object-level to the meta-level, information needed by BOLERO in order to evaluate the state of the problem solution at the object-level and to be able to plan in consequence. That is, they map the input facts and goals from the object-level to the meta-level. The downwards reflection rule, R_d shows how the decisions taken at the meta-level have their consequences at the object-level.That is, it indicates that a plan generated by BOLERO modifies the plan of the RBS interpreter: maps a sequence of actions into a sequence of goals of the object-level.

At any moment, BOLERO-RBS is active in only one of the two levels, either in the meta-level (BOLERO) or in the object-level (RBS). The active level is called *locus of action* (van Harmelen, 1991). When a user wants to solve a problem, the first locus of action is BOLERO at its initial state σ_{BOLERO_0}. There exists a default plan p_0 in σ_{BOLERO_0} that starts the RBS. We represent by σ_{RBS_0} the state when the RBS system is activated the first time. Following the downwards reflection rule R_d applied to the default plan p_0 RBS has an initial set of goals to start (meta-level reflect down). The locus of action is now RBS, the interpreter of which begins to pursue the first goal registered in $\lfloor p_0 \rfloor$ in the initial state σ_{RBS_0}. During the reasoning process RBS can achieve, in a given state σ_{RBS_1}, a new fact f (by acquiring an input fact from the user or by deducing a goal fact). Then, the new fact $\lceil f \rceil$ is also known by BOLERO through the upwards reflection rules R_{u_i} or R_{u_g} and the new locus of action is now BOLERO in the state σ_{BOLERO_1} (meta-level lift up). All facts known up to σ_{BOLERO_1} (including $\lceil f \rceil$), lead BOLERO to generate a new plan in the state σ_{BOLERO_2}. This new plan modifies the goal sequence $\lfloor p_1 \rfloor$ that the RBS interpreter was trying to satisfy in σ_{RBS_1}, in a new sequence in the state σ_{RBS_2} (meta-level reflect down). RBS is then the new locus of action. This process continues until BOLERO decides a solution has been achieved and execution is terminated.

To illustrate with an example of the sequence of loci of action changes that holds in the solution of a problem let's assume that we want to know which is the agent causing the pneumonia of patient *r13v1*. The patient's initial data are: *(age, 79)*, *(community-acquired, yes)*, *(severity, serious)*, *(sex, woman)*, *(asylum, unknown)*, *(pathological-records-of-interest, diabetes)*. The sequence of loci of action changes that take place in BOLERO-RBS is the following:

σ_{BOLERO_0}: BOLERO generates the initial plan p_0 = *[gather-information]*.

σ_{RBS_0}: *RBS* starts pursuing the goal *gather-information* (p = $\lfloor$ *[gather-information]* $\rfloor$).

σ_{RBS_1}: During the reasoning process, *RBS* asks the user the patient's age, and the fact *(age, 79)* is obtained.

σ_{BOLERO_1}: The new information $\lceil$*(age , 79)*$\rceil$ is not relevant in BOLERO from the strategic point of view, that is, in order to generate a new plan.

σ_{BOLERO_2}: BOLERO keeps on the plan *[gather-information]*

σ_{RBS_2}: *RBS* goes on with the goal *gather-information.*

...

σ_{RBS_7}: The *RBS* interpreter acquires the fact *(pathological-record-of-interest, diabetes)* from the user.

σ_{BOLERO_7}: The fact $\lceil$*(pathological-record-of-interest, diabetes)*$\rceil$ is a relevant piece of information that enlightens the diagnostic process of the patient.

σ_{BOLERO_8}: Knowing $\lceil$*(pathological-record-of-interest, diabetes)*$\rceil$, and summing up the rest of the information that has been gathered since the beginning of the execution (i.e. $\lceil$*(age, 79)*$\rceil$, $\lceil$*(community-acquired, yes)*$\rceil$, $\lceil$*(severity, serious)*$\rceil$, $\lceil$*(sex, woman)*$\rceil$, $\lceil$*(asylum, unknown)*$\rceil$ *)*, BOLERO builds a new plan: *[bact-atyp, pneumococcus, legionella, s-pyogenes]*.

σ_{RBS_8}: The RBS interpreter has a new list of goals to pursue, p = $\lfloor$ *[bact-atyp, pneumococcus, legionella, s-pyogenes]*$\rfloor$, and starts pursuing *bact-atyp*.

$\cdots$ And so forth until at the n-th locus of action changes. When the diagnoses $\lceil$*(pneumococcus, sure)*$\rceil$ and $\lceil$*(legionella, somewhat-possible)*$\rceil$ are known, BOLERO decides that the current problem solving state of the RBS is a goal state and stops the execution.

Whenever new information is known (i.e. an input or goal fact), the meta-level checks if it is possible to generate a new plan able to adapt the reasoning process to the new circumstances. If a new plan is generated it is executed at once; otherwise the old plan goes on. The integration of BOLERO and the RBS allows the system to react to information changes by shifting from the object to the meta-level and re-planning.

It is interesting to note, first, that each generated plan may come from different past cases. That is to say, BOLERO performs case-based planning with multiple cases. Second, in BOLERO-RBS we are improving the problem-solving efficiency by means of an strategic planning of the goals of the RBS. Third, BOLERO can learn from its own experience: when a problem is solved in BOLERO-RBS, a new case is automatically produced. The new case, as any other case in BOLERO, is composed by all the facts gathered during problem solving plus the final plan. Then BOLERO can learn from its own experience from this new case, as we explain later in the paper.

4 BOLERO: Learning and generating plans

As with any case-based system, in order to solve a problem BOLERO retrieves cases from memory and adapts the solution (plan) of the retrieved case to the current situation. If the system has no cases in memory, however, it will fail to solve any problem. Therefore, a preliminary learning phase, that we have called learning from a teacher, is necessary. In this training stage, cases provided by a teacher are organized in memory by means of a generalization procedure in such a way that a case can be indexed by its own features as well as by their generalizations. After that phase, methods for retrieval and adaptation can be applied to reuse cases in novel situations. Once a problem is solved, BOLERO has a new experience in building plans, that is, a new case. The evaluation of the case leads to knowledge of the success or failure of the plan generated by BOLERO, and from the results of the evaluation BOLERO can learn from its own experience. The solution of any new case always causes a memory reorganization: when cases are successfully solved, they are incorporated into the memory (and so new generalizations can be performed); when cases are incorrectly solved, they are used to tune the memory organization and the correct case is obtained from to the teacher and incorporated also in memory. Therefore, BOLERO increases its knowledge about plans monotonically, as other CBR systems do (Compton et al., 1992).

We continue this section by describing the learning from a teacher, retrieval, adaptation, evaluation and learning from experience methods.

4.1 Learning plans from a teacher

Whereas most of the current case-based systems assume that the system has in its memory a set of past experiences or past cases, we have distinguished in BOLERO an initial stage in which the system learns from scratch. If the system has no planning knowledge, when trying to solve a problem in a domain where n actions are possible, the system would build a unique plan p containing all the actions and the system would be inefficient. However, if we first train our system with a few set of cases, when solving a problem we can reuse the most similar previous (training) plan and avoid applying the overall set of actions. Moreover, distinguishing a training phase harmonizes with the belief that a quite organized training phase followed by a practice phase where complex problems are solved, leads to efficient problem solving.

In BOLERO learning from a teacher consists of organizing in memory a set of cases provided by a teacher. The organization of cases must favor the re-use of plans. That is, we need an organization that relates the plans with the situations in which they have been executed. For this reason we have decided on a tree organization of cases where leaf nodes correspond to complete plans and where non-leaf nodes correspond to partial plans. This organization is easily achieved by taking advantage of the precedence relation of plans and situations of episodes of cases described in section 2.4. Assuming that any plan begins with an initial action a_0, and using the fact that a plan p^i_j of a given case C^i has only one action more that its predecessor p^i_{j-1}, we get a hierarchy of plans. Each plan of a node of the hierarchy corresponds to a case or more. That is, different episodes of different cases may have the same plan. However, the situations of two episodes with the same plan can be different. So, together with a plan p_i, in each node N^i of the hierarchy we keep a generalized situation v^i that stands for all the situations of all the episodes with the same plan. That is, $N^i = \langle v^i, p^i \rangle$. A generalized situation of a node N^i is used to index the plan p^i of the node.

Once known the organization of cases in memory, learning from a teacher can be easily defined as an incremental process in which we distinguish two basic issues: incorporation of a case in memory and generalization of situations.

4.1.1 Top-down, incremental case incorporation

Learning is incremental: at the beginning the memory is empty, and the training cases are incorporated one by one. The incorporation of a new case results in a new memory organization. Incremental learning allows us at any time to refine BOLERO planning knowledge by means of the incorporation of a new case in memory.

The incorporation process of a case C^i into memory consists of going down from the top of the plan memory, by comparing the plans of the episodes of C^i with the plans of the nodes (see algorithm in table 1). That is, we compare the plan p^i_j of the episode δ^i_j with any node of the level j of the hierarchy. If there exists a node N^k in the level j that has a plan $p^k = p^i_j$, then the generalized situation of N^k is updated (see next section). The incorporation process of C^i proceeds with the next episode δ^i_{j+1} and the nodes of level $j+1$ that are successors of N^k. This process is performed in each episode of the case. If no node is found that has a plan equal to p^i_j, then a new branch in the level $j-1$ of the hierarchy is opened, and the episodes $\delta^i_j, \ldots, \delta^i_{n^i}$ become new nodes of that branch. The memory organization that results from the incorporation of cases *b06v1* and *g01v1* of figure 3 is shown in figure 6.

4.1.2 Generalization

Each node of the hierarchy has attached a set of generalized facts that represent a generalized situation. Generalized situations are the result of a generalization process among facts of different situations that belong to episodes with the same plan. For example, in figure 6 the generalized fact *(severity, { somewhat-serious, light})* is the result of the generalization of the fact *(severity, somewhat-serious)* of case *b06v1* and the fact *(severity, light)* of case *b01v1*.

Any plan of a node is indexed by the facts that compose the generalized situation. A generalized fact, g^i_j, of a generalized situation v^i is represented by a tuple $\langle at^i_j, v^i_j, w^i_j, \sigma^i_j \rangle$ where at^i_j is an attribute, v^i_j the value, w^i_j the strategic relevance degree, and σ^i_j the frequency. Strategic relevance degree represents the weight of a generalized fact when used as an index to the plan of the node. A generalized fact with an strategic relevance degree 0 means that it is irrelevant when building the plan of the node while an strategic relevance degree of 1 means that the generalized fact is definitely relevant. The strategic relevance is explicitly represented in the nodes since it can be tuned by learning from experience, as explained in section 4.4.2. The strategic relevance degree plays an important role in the retrieval process, and differentiates BOLERO's learning method from other machine learning systems which rely on frequency of facts (COBWEB (Gennari et al., 1990),

Table 1: Algorithm to incorporate a case in memory. Input: a new case and a memory of cases. Output: an updated memory of cases

```
incorporate(C:case,N:root)
      ; case: $C = [\delta_0, \ldots, \delta_n], \delta_i = \langle s_i, p_i \rangle$
      ; node: $N = \langle v, p \rangle$
1.    i=0;
2.    While $i < n$
3.          If $\exists N' \in successors(N)$ such that $p' = p_i$
4.          Then  $v = generalize(v', v_i)$
5.                $N = N'$
6.                $i = i + 1$
7.          Otherwise    $N = built\text{-}nodes([\delta_i, \ldots, \delta_n], N)$
8.                       $i = n$

built-nodes($\Sigma$:episodes, N:node)
      ; $\Sigma = [\delta_i, \ldots, \delta_n]$
1.    k=i
2.    While $k < n$
3.          $N'$ = new-node
4.          $successors(N) = successors(N) \cup \{N'\}$
5.          $v' = s_k$
6.          $p' = p_k$
7.          $k = k + 1$
```

LAMDA (Piera et al., 1989)). The indexing mechanism of those systems rely on how often facts occur, but in medical diagnosis we found that a fact with a low frequency can be quite relevant[5].

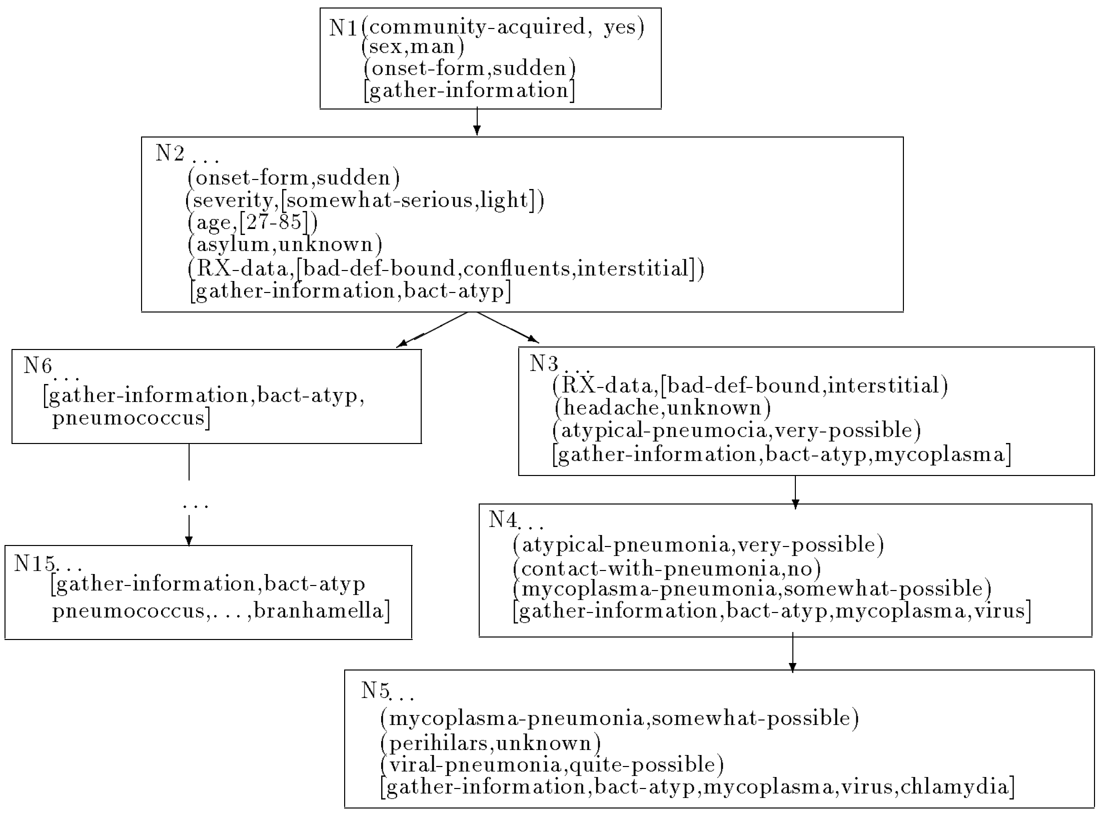
Generalized facts are built on the incremental generalization procedure. Given a generalized situation v^i of a node, and a situation s_j^k of the episode δ_j^k of a case, the generalization process computes the new generalized situation v^i in the following steps:

1. Perform a value generalization among the facts of s_j^k and the generalized facts g_l^i of v^i with the same attribute (i.e. facts in $v^i \cap s_j^k$, where the intersection operation is done referring to the attributes).
2. Among the rest of the facts (i.e. $s_j^k \setminus v^i$):
   (a) Do an abstraction, if possible.
   (b) Otherwise, add the fact of s_j^i to v^i as a generalized fact.

In the generalization process we distinguish value generalization and abstraction. On the one hand, the result of the generalization of two values, v_1 and v_2, is the union of both, $\{v_1\} \cup \{v_2\}$. On the other hand, abstraction is based on the relations among attributes expressed in the domain network. Given a set of attributes, $at_1, \ldots, at_n$, abstraction produces an unique attribute at. This attribute at has a relation with any of the attributes at_i inside the domain network. For example, and given the domain network of figure 1, the abstraction of the facts *(circulatory-data-assoc, pericarditis)*, *(neurological-data-assoc, encephalitis)*, and *(skin-data-assoc, cellulitis)* is the fact *(clinical-data-assoc, { circulatory-data-assoc, neurological-data-assoc, skin-data-assoc })*.

Finally we want to stress the importance of the last point of the generalization procedure: the addition of facts. As a consequence, the generalization process builds disjunctive descriptions of plans. In this sense, we argue that disjunctive characterizations help to face the *overfitting* problem that most of the conjunctive algorithms suffer (Belew, 1992). The overfitting problem relates the number of training cases with the predictivity of the description learned (and so, the probability that future examples of the same concept fit in the learned description). Conjunctive algorithms

[5]This has also been found in EUREKA (Elio & Scharf, 1990).



N1(community-acquired, yes)
(sex,man)
(onset-form,sudden)
[gather-information]
N2 . . .
(onset-form,sudden)
(severity,[somewhat-serious,light])
(age,[27-85])
(asylum,unknown)
(RX-data,[bad-def-bound,confluents,interstitial])
[gather-information,bact-atyp]
N6 . . .
[gather-information,bact-atyp,
pneumococcus]
. . .
N15. . .
[gather-information,bact-atyp
pneumococcus,. . . ,branhamella]
N3 . . .
(RX-data,[bad-def-bound,interstitial)
(headache,unknown)
(atypical-pneumocia,very-possible)
[gather-information,bact-atyp,mycoplasma]
N4. . .
(atypical-pneumonia,very-possible)
(contact-with-pneumonia,no)
(mycoplasma-pneumonia,somewhat-possible)
[gather-information,bact-atyp,mycoplasma,virus]
N5. . .
(mycoplasma-pneumonia,somewhat-possible)
(perihilars,unknown)
(viral-pneumonia,quite-possible)
[gather-information,bact-atyp,mycoplasma,virus,chlamydia]

Figure 6: Memory organization after learning case b06v1 and case g01v1 of figure 3 from scratch.

(i.e. algorithms that only take into account common features (Pazzani & Sarrett, 1992)) tend to be less predictive when the number of cases increases.

4.2 Plan generation

In a dynamic environment, acquired data is incomplete and there is a need to develop a technique that mitigates the missing information. In this sense, BOLERO is able to generate a plan anytime according to the available data. If the information about a problem changes after applying the plan, BOLERO can adapt the reasoning process by generating a new plan.

BOLERO's method of building a plan, as in any other case-based system, consists of two basic steps: retrieval and adaptation. When any new fact is known, this fact is lifted up from the object-level RBS to the meta-level and BOLERO retrieves from memory the plan most adequate to the current situation (see figure 4). If the retrieved case is the same one retrieved for the previous situation, no more reasoning is performed. Otherwise, the plan of the retrieved case is adapted to the current situation and then lifted down to the RBS for its execution.

4.2.1 Plan retrieval

Given a situation, plans are retrieved from memory by using a spreading activation mechanism. For each retrieved node its matching degree with the current situations is computed by means of a similarity measure. Once the retrieved node with the maximum matching degree is known, BOLERO proceeds to adapt the plan of the retrieved node. If it happens to be that no nodes are retrieved, or that the plans of the retrieved nodes have already been executed as a consequence of the adaptive planning BOLERO then decides to terminate problem solving. These situations usually happen when there are few cases in memory.[6]

Spreading activation. Let us suppose that we are trying to diagnose a patient in BOLERO-RBS, and the current set of known facts is s_a. It is possible to retrieve nodes from BOLERO's memory by means of a constrained spreading activation that uses the facts of s_a as indexes. First, any attribute of the domain network that has some relation with any attribute of a fact in s_a will be activated. The activation is constrained in the sense that not all the relations can be used. For example, if an attribute at_1 activates an attribute at_2 because of a *kind-of* relation (i.e. at_1 is a kind of at_2), it is not possible to activate a third attribute f_3 because it has a specialization relation with f_2 (i.e. at_3 is a specialization of at_2). Secondly, any activated attribute in the domain network is used as an index to activate nodes with generalized facts that have those attributes.

Similarity metric. We calculate the degree of similarity of the nodes retrieved in relation to s_a, based on two matching functions, m_f and m_s, defined in the unit interval . Function m_f computes the similarity between a fact f_a in s_a and a generalized fact g^i_j in a node N^i. If f_a and g^i_j have the same attribute (otherwise $m_f(g^i_j, f_a) = 0$), then

$$m_f(g^i_j, f_a) = \begin{cases} 1 & if \;\; (monovalued(f_a) \wedge (v_a = v^i_j)) \vee (multivalued(f_a) \wedge (V_a \cap V^i_j \neq \emptyset)) \\ 1 - d(v_a, v^i_j) & if \;\; fuzzy(f_a) \\ 0 & otherwise \end{cases}$$

where $d(v_a, v^i_j)$ is a function defined in [0,1] that measures the distance between two fuzzy values similar to the one defined in section 2.4 (see also (López, 1993)). Fuzzy values are ordered and their relative distance can be established. Set-valued facts can be compared by a membership relation that outputs 1 (the value belongs to the set of possible values) or 0 (otherwise).

The similarity function m_s combines the evaluation of m_f for all the generalized facts g^i_j of a generalized situation v^i in two steps. Firstly, we compute the weighted addition of the similarities among facts, as follows:

$$Sum(v^i, s_a) = \sum_{g^i_j \in v^i} W(g^i_j, f_a) m_f(g^i_j, f_a)$$

where W is the strategic relevance computed for g^i_j in s_a. The evaluation of W is the following:

$$W(g^i_j, f_a) = \begin{cases} w^i_j & if \;\; monovalued(f_a) \\ C_n(\{w_l | v_l \in V^i_j \cap V_a\}) & if \;\; multivalued(f_a) \\ 0 & otherwise \end{cases}$$

where w^i_j is the strategic relevance degree of g^i_j; w_l the strategic relevance degree of the value v_l of g^i_j; and C_n is a disjunctive aggregation function of n evidences based on a t-conorm (Piera et al., 1989; Dubois & Koning, 1991; Bonissone & Decker, 1986). Particularly, we have chosen $C(x_1, \ldots, x_n) = max(x_1, \ldots, x_n)$. Secondly, and based on $Sum(v^i, s_a)$, we calculate m_s as a mean between the inclusion degree and exclusion degree of s_a in a generalized situation v^i The inclusion degree of s_a in v^i is

$$I_{m_s}(v^i, s_a) = \frac{Sum(v^i, s_a)}{n + k}$$

where n is the number of generalized facts in v^i that have a frequency equal to 1; and k is the number of generalized facts in v^i that are in s_a (i.e. activated) and whose frequency is less that 1. The inclusion degree relates the number of activated facts in a generalized situation v^i with

[6]Note that with few cases in memory BOLERO can only recover few plans and those cannot be suitable for the current problem (i.e. they are not activated for the facts of the new problem).

respect to the number of facts (n) that are expected to be present in the current situation in order to apply the plan associated to the node that v^i indexes.

Analogously, the exclusion degree relates the number of activated facts in v^i with respect to the total number of facts in the current situation s_a. It is defined as follows:

$$E_{m_s}(v^i, s_a) = \frac{Sum(v^i, s_a)}{|s_a|}$$

where $|s_a|$ is the cardinality of s_a. Finally, the similarity function combines both:

$$m_s(v^i, s_a) = \frac{I_{m_s}(v^i, s_a) + E_{m_s}(v^i, s_a)}{2}$$

It is interesting to note that the strategic relevance degree of facts, w_i, plays an important role in the evaluation of m_s as will be shown experimentally in section 6. And it is also interesting to see that m_s allows us to recognize a generalized situation v^i identical to s_a, since in this case $m_s(v^i, s_a) = 1$.

We want to stress that the matching process is incrementally performed: initially there are no known facts. For any new fact known in the environment as a consequence of the execution of a previous plan in BOLERO-RBS, activation is performed and the degree of matching re-computed for the recent re-activated nodes.

4.2.2 Case adaptation

The last step in the plan generation process consists of adapting to the current situation the plan of the node retrieved with the highest similarity degree[7]. The adaptation consists of avoiding the repetition of actions that have already been executed. Since situations are assumed to be monotonic, the execution of an action already executed will have the same effects. The plan of the retrieved node is adapted and the resulting plan, $p = [a_0, \ldots, a_i]$ is a new plan with which to continue the diagnosis and whose actions $a_0, \ldots, a_i$, have not been executed. The algorithm is shown in table 2.

Table 2: Algorithm to adapt a plan to the current situation.

```
adapt(N:node,C:case)
        ; current case: $C = [\delta_0, \ldots, \delta_n]$
        ; $\delta_i = \langle s_i, p_i \rangle$ for $i < n$, $\delta_n = \langle s_i \rangle$
        ; $p_{n-1}$ is the plan conformed by the currently executed actions
        ; $s_n$ is the current situation $s_a$.
        ; node: $N = \langle v, p \rangle$, $p = [a_0, \ldots a_m]$
1.      $j = 0$; $p_a = p$;
2.      While $j < m$
3.              If $a_j \in p_{n-1}$
4.              Then  $p_a = p_a \setminus \{a_j\}$
5.      return($p_a$)
```

As a consequence of adaptation BOLERO may generate a new plan while solving a new problem. A plan is new in the sense that it does not exists such a plan in the plan memory before being generated by the system. The new plan is the result of the adaptation of parts of plans (subplans) of different nodes retrieved along the problem solving process of a case in BOLERO-RBS. For example, let us assume the plan memory of figure 7a and that in a given moment, BOLERO retrieves the node n_5 to solve a problem C according to the retrieval function described in previous section. Then, the plan associated with n_5, *[gather-information, bact-atyp,*

[7]In case of having more than one node with the same similarity degree, we have also developed another kind of heuristic adaptation rules (as for example: choose common actions of plans) that can be found in (López, 1993).

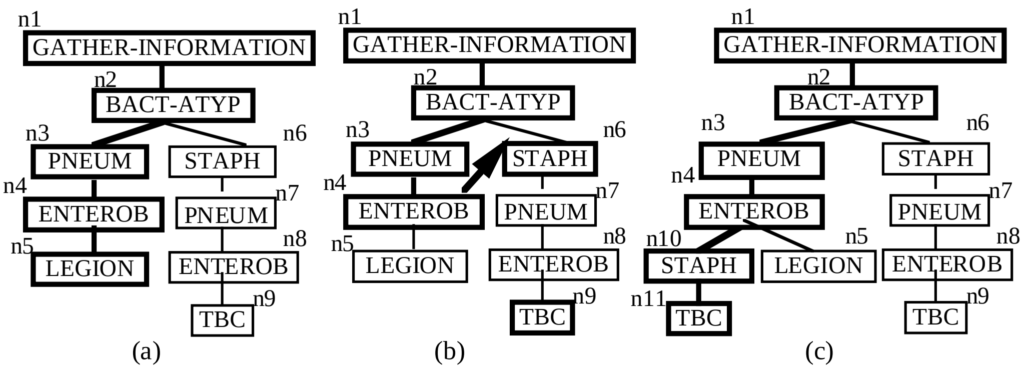


Figure 7: Schematic representation of the solution of a case. To simplify, in each node it is only shown the action of that which differentiates the plan of the node with the plan of its parent. (a) The plan generated is based on the plan of node n_5. (b) The plan generated is based on the plan of node n_9. (c) Plan memory after the incorporation of the new case.

pneumococcus, enterobact, legionella], is selected. However, after applying action *enterobact*, the availability of new information allows BOLERO to retrieve node n_9 (figure 7b) with a higher degree of similarity than node n_5. The plan of the retrieved node n_9 is adapted giving the following current plan to execute: *[staphylococ, tuberculosis]*. This plan *[staphylococ, tuberculosis]* is a new one (i.e. does not exist previously in memory). If BOLERO arrives at a goal state after applying this adapted plan, then the final plan that has led to the solution of the problem is: *[gather-information, bact-atyp, pneumococcus, enterobact, staphylococ, tuberculosis]*. Innovative plans are constructed from partial plans because (1) plan generation and execution are interleaved (in the meta-level architecture) and (2) because replanning (and plan adaptation) can be performed at anytime.

4.3 Case evaluation

The goal of evaluation is to know the success or failure of plans. In order to carry out evaluation we need additional knowledge that allows us to evaluate a behavior as correct or wrong, and eventually, compute the degree of correctness and accuracy. Such knowledge is expressed in the form of a gold standard. The gold standard is the right solution, the expected or wished one. It is a model of functionality that registers the domain solutions (i.e. the diagnosis) as well as the optimal plan (that is, the sequence of actions that leads to the achievement of the diagnosis).

The difficulties of determining such a gold standard have been already shown elsewhere (López, 1991), and are emphasized in a medical domain where the solution of a problem may be not unique. The approximation taken in BOLERO has been to define an evaluation standard (ES) composed by an admissible plan ES_p and a set of domain solutions ES_s. The ES can be acquired either from the expert that provides the cases or by several experts, establishing the solution to a problem by consensus. In this sense, although evaluation is automatic, it requires the participation of at least a human expert that provides the solution to the problem from his point of view. We defined a ES for every case.

With the ES we have defined three measures of plan evaluation: degree of success, degree of focus and degree of disorder. Those measures are applied to the final plan p that being executed from the initial situation leads to the solution of the problem. The degree of success relates the number of actions in p that are in ES_p with the total number of actions in ES_p. Such a measure guarantees that all the actions that the physician proposes have also been performed by the case-based planner. At the same time, by using such a measure, we are introducing a conservative bias into the system, since we are forcing that the plan contains all the actions of ES_p. This bias agrees with the physicians' policy when diagnosing a patient: they prefer to perform additional tests and examinations rather than forgetting to consider some dangerous disease.

The degree of focus is easily defined as the contrary of the degree of out-focus. The degree of out-focus relates the number of actions in a plan p that can be saved with respect to the total number of actions of p. An action can be saved if it is neither in ES_p nor offers evidence to some diagnoses in ES_s. Performing extra actions means to waste resources. Including spurious actions in a plan degrades the degree of focus of BOLERO.

The degree of disorder relates the number of actions that are in an adequate order with respect to the total number of actions in p. The correct order of an action is determined by the degree of evidence with which its execution contributes to the solution of the problem (see (López, 1993)). Actions that offer more evidence should ideally be performed first.

To sum up, we say that a plan succeeds completely if the degree of success is 100%, the degree of focus is 100%, and the degree of disorder is 0%. Otherwise we say that the plan is imperfect. For example, let us suppose that BOLERO has solved a case with the plan p=*[gather-information, bact-atyp, mycoplasma, chlamydiae, q-fever]*, and that the ES_p is *[gather-information, bact-atyp, virus, mycoplasma, chlamydiae]*. The degree of success of p is 80% since the plan does not contain the action *virus* and so four of the five total actions required to solve the problem have been correct. The degree of out-focus is 20% since one action, *q-fever*, from the five actions that compounds p is not in ES_p. Besides, a deeper checking shows that *q-fever* does not offer evidence for any of the diagnoses, since the result of its execution has been the fact *(pneumonia-by-q-fever, unknown)*. Consistently, the degree of focus is 80%.

4.4 Learning from experience

We know from the evaluation method whether the plans have been completely successful or, on the contrary, imperfect. Thus, BOLERO can learn from successful plans and from imperfect plans.

4.4.1 Learning from successful plans

A successful plan can be either a plan already in memory or a new plan built by BOLERO as a consequence of re-planning. That is, a final plan built to solve a problem is the sequence of parts of plans that BOLERO has generated and that have been effectively executed by the RBS. For example, we have seen that BOLERO solves a problem with the plans of nodes n_5 and n_9 of figure 7. The resulting plan is *[gather-information, bact-atyp, pneumococcus, enterobact, staphylococcus, tuberculosis]*. This new plan is not in memory, and BOLERO can learn it (figure 7c). Then, learning from successful plans means incorporating new plans in memory, following the procedure described in 4.1. The results of learning from successful plans are then new generalization of indexes and a more refined plan memory.

4.4.2 Learning from imperfect plans

When a plan is imperfect, BOLERO has an opportunity to learn. First, the expert gives the correct solution (i.e. a new case), that can be incorporated in memory and so, BOLERO learns new plans. Secondly, given that the indexing mechanism used to recover that plan has been incorrect, we need to tune the indexes in order to avoid retrieving the same nodes in the same circumstances. Retrieval is based on the matching functions m_f and m_s, and particularly m_s closely depends on the strategic relevance degree of the generalized facts attached to nodes. So, one way of modifying the indexing mechanism is to update the strategic relevance degree of the generalized facts.

We distinguish three main steps in order to learn the strategic relevance degree of facts: 1) determination of the set of nodes that should and should not have been retrieved; 2) distinction of generalized facts that caused the wrong retrieval; and 3) actualization of the strategic relevance degree of generalized facts. The algorithm is shown in table 3.

Table 3: Algorithm to learn the strategic relevance degree of facts.

```
learn-w(C:case;$\mathcal{N}$:nodes;$p_c$:plan)
        ; case: $C = [\delta_0, \ldots, \delta_n]$, $\delta_i = \langle s_i, p_i \rangle$, $s_i = \{f_i^j\}$
        ; retrieved nodes along problem solving: $\mathcal{N}$
        ; node: $N_i = \langle \upsilon_i, p_i \rangle$, $\upsilon_i = \{g_i^j\}$
        ; correct plan: $p_c$
1.      $N_c$ = select-correct-nodes($p_c$)
2.      $N_e = \mathcal{N} \setminus N_c$
3.      label-facts($\{f | f \in \upsilon_i, N^i \in N_c\}$, $\{f | f \in s_i, s_i \in C\}$, $\{f | f \in \upsilon_i, N^i \in N_e\}$)
4.      For each $N_i \in N_c \cup N_e$
5.              For each $g_i^j \in \upsilon_i$ update-w($g_i^j$)

select-correct-nodes($p_c$:plan)
1.      If $\exists N$ such that $p_c = p$
2.      Then $return(\{N\})$
3.      Else $return(\{N | p \sim p_c\})$

label-facts($F_c$:facts,$C : facts$,$F_e$:facts)
1.      For each $f \in F_c$
2.              If $f \in C$ and $f \notin F_e$
3.              Then $f$ is *type-II*
4.              Else $f$ is *type-III*
5.      For each $f \in F_e$
6.              If $f \in C$ and $f \notin F_c$
7.              Then $f$ is *type-I*
8.              Else $f$ is *type-III*
```

Determination of the set of nodes that should and should not have been retrieved. Let's call p_c the plan that the teacher provides as the correct one and p the imperfect plan generated by BOLERO; and analogously let's call $\mathcal{N}_c$ to the set of nodes that should be retrieved, and $\mathcal{N}_e$ the set of nodes that shouldn't. Then,

- the set $\mathcal{N}_c$ is composed by an unique node whose plan is p_c or, failing it, by the nodes whose plan is similar to p_c (for further details see (López, 1993)).
- Since the plan p is either an existing plan in memory or the result of the adaptation of different plans of different retrieved nodes $\mathcal{N} = \{N^1, N^2, \ldots, N^n\}$, the set $\mathcal{N}_e$ is composed by the retrieved nodes $\mathcal{N}$ except the nodes that also belong to $\mathcal{N}_c$ (i.e. $\mathcal{N}_e = \mathcal{N} \setminus \mathcal{N}_c$).

Distinction of generalized facts that cause the bad retrieval. Having computed $\mathcal{N}_e$ and $\mathcal{N}_c$, it is possible to distinguish the following set of generalized facts:

- Type–I: Generalized facts used to index to nodes in $\mathcal{N}_e$ and that are not in nodes in $\mathcal{N}_c$.
- Type–II: Generalized facts attached to nodes in $\mathcal{N}_c$ and that are not in nodes in $\mathcal{N}_e$.
- Type–III: Generalized facts that index nodes in $\mathcal{N}_e$ and nodes in $\mathcal{N}_c$.

Updating the strategic relevance degree of generalized facts. Once we have determined the three types of generalized facts, updating the strategic relevance of the generalized facts consists of: 1) decreasing the strategic relevance degree of generalized facts of type–I; 2) increasing the strategic relevance degree of generalized facts of type–II; and 3) leaving untouched the strategic relevance degree of generalized facts of type–III[8].

We increase and decrease the strategic relevance degree of a generalized fact g_i by using the following evidence combination function:

[8]Note that type–III facts does not provide information about incorrect retrieved nodes.

$$\textit{update-w}(g_i) = \begin{cases} T(1-n, w_i) & \textit{if } g_i \textit{ belongs to type-I} \\ C(w_i, T(n, w_i)) & \textit{if } g_i \textit{ belongs to type-II} \end{cases}$$

where n is a parameter ($n \in [0,1]$) that regulates the suddenness of the updating [9]; w_i is the strategic relevance degree of g_i; and $T(x,y)$ is a t-norm and $C(x,y)$ is a t-conorm dual to $T(x,y)$ (see (Bonissone & Decker, 1986)) that are the functions we use to combine the evidence found about w_i. Particularly, we have chosen: $C(x,y) = x + y - xy$ and $T(x,y) = xy$.

To end this section, we want to stress that all the learning from experience is performed automatically. The expert or experts are only required to provide the correct solution during the evaluation stage.

5 BOLERO: Learning and retrieving goal states

Planning in medical domains requires a method to recognize when a solution to a problem has been achieved, i.e. when the system has reached a goal state. For this purpose, the physician can provide a set of conditions that identifies goal states. We call them *termination conditions* since recognizing a goal state has the effect of ending the problem solving process. Such conditions take a form like: "if pneumonia caused by pneumococcus is deduced with a certainty degree higher than possible, then the diagnosis is definitive". However, the information known about a patient is so uncertain and incomplete that seldom a fact is deduced with enough certainty in order to satisfy any termination condition. Therefore, when solving a problem, if no termination condition is satisfied, the solution of a problem in BOLERO-RBS will be large and exhaustive. For this reason we have developed additional methods in BOLERO to learn and recognize goal states and so to decide about problem-solving termination.

For example, in a given moment BOLERO knows, among other patient's data, the following diagnoses {*(atypical-pneumonia, possible)*, *(mycoplasma-pneumonia, quite-possible)*, and *(chlamydiae-spp, possible)* }. None of the diagnoses goes beyond the certainty degree that the physician has defined in the termination conditions. Should BOLERO continue trying to prove more possible diagnosis? It seems a good idea to use past cases, past experiences to decide if the current problem-solving state is a goal state. In fact, we speak of case-based learning of goal states methods that run concurrently with the methods of learning and generation of plans of the previous section.

Case-based learning of goal states is the third method that BOLERO can use to decide upon problem solving termination. That is, BOLERO can stop problem solving: (1) when there are no retrieved nodes from memory that contribute plans with which proceed problem solving (see section 4.2.1); (2) when a termination condition is satisfied; and (3) when a past goal state retrieved from memory matches with a certain degree the current problem solving state. In this section we focus on the description of method (3), the case-based reasoning on goal states. Again, as in any case-based method, we distinguish learning from retrieval.

5.1 Learning goal states

Cases with the same final plan can have different goal states. Particularly, facts that constitute the domain solution of a problem (i.e. diagnoses or goal facts F_g) are different in each goal state since they are achieved applying the actions of the plan and the results of each action depend on the known facts. However, although domain solutions are not identical for cases with the same plan, it is reasonable to think that they will be similar. For this reason, when incorporating a new case in memory, we store in leaf nodes, together with the final plan p^i and the generalized situation v^i, the set of domain solutions $\mathcal{S}^i = \{S^i_j\}$ of all problems solved with plan p^i. A domain solution S^i_j of a past case C^j stored in a node N^i (i.e. $S^i_j \in \mathcal{S}^i$) is a collection of goal facts and it is included in the goal state of the case. That is, if $C^j = [\delta^j_0, \ldots, \delta^j_n]$, where $\delta^j_k = \langle s^j_n, p^j_n \rangle$ and s^j_n is

[9] We have used small values, following a parsimonia criterion and experimentally n has been set up to 0.2.

the goal state of C^j, then $S^i_j = \{f | f \in F_g\}$ is the domain solution of C^j and $S^i_j \subset s^j_n$. Analogously we say that the generalized situation v^i of the leaf node is a generalized goal state, and $\mathcal{S}^i$ contains all past domain solutions achieved with the plan p^i.

Any problem-solving state containing a domain solution similar to a past one and stored in a leaf node will be recognized by BOLERO as a goal state, as explained in the next section.

5.2 Goal state retrieval

In order to decide whether or not the current problem solving state in the RBS has reached a goal state (a domain solution), BOLERO uses two different strategies of goal state retrieval. The first one consists on retrieving from memory a past goal state similar to the current problem solving state. The second one, that we call dissimilarity detection, evaluates the dissimilarities between situations that index already executed plans of past goal states and situations that retrieve plans likely to be executed.

5.2.1 Past goal state recognition

When a leaf node N^i is retrieved in a given problem-solving situation s_a, the set of domain solutions $\mathcal{S}^i$ attached to that node is also retrieved. Then, if the plan p_i of that node N^i has already been executed, BOLERO computes the similarity of the domain solution S_{s_a} of the current problem solving state s_a to every domain solution S^i_j in $\mathcal{S}^i$. If the similarity is considered sufficient, the current state s_a is recognized as a goal state. Consistently, BOLERO stops problem-solving.

For a current state s_a, the domain solution is $S_{s_a} = \{f | f \in s_a \wedge f \in F_g\}$. The similarity of two domain solutions is the following:

$$sim(S_{s_a}, S^i_j) = 1 - d_e(S_{s_a}, S^i_j)$$

where S^i_j is the domain solution of a past case; and $d_e(S_{s_a}, S^i_j)$ the reduced Euclidean distance defined between two domain solutions. The reduced Euclidean distance has been defined as follows:

$$d_e(S_{s_a}, S^i_j) = \sqrt{\frac{1}{n} \sum_{m=1}^{n} d(f^a_m, f^j_m)^2}$$

where $n = |F_g|$; and $d(f^a_m, f^j_m)$ the distance defined between two fuzzy facts[10]. Then, if the similarity is greater or equal than a given threshold[11], $sim(S_{s_a}, S^i_j) \geq \xi$, problem solving can be stopped, because BOLERO recognizes the current state s_a as a goal state.

The definition of the reduced Euclidean distance among domain solutions, although successfully used in expert system evaluation in our Institute (Belmonte, 1990; Verdaguer, 1989), is a weak procedure in order to recognize a goal state. The great amount of different goal states, even for a given plan, lessens the likelihood to find similar goal states in memory. There are more goal states than possible plans. So, we have completed the procedure of goal state retrieval with the procedure that follows.

5.2.2 Dissimilarity detection

At any given moment, a final plan stored in a leaf node has been fully executed. If past goal state recognition does not terminate problem solving, BOLERO can proceed by reusing the plans of other retrieved nodes. For example, it can continue by selecting the node with the second highest similarity degree (the node with the highest similarity degree is the one whose plan has been just executed). It is possible to think of continuing problem solving if the highest similarity degree is

[10] Since fuzzy facts can be compared according to their position inside a set of linguistic labels (see section 2.4) the computation of the reduced euclidean distance becomes rather inexpensive.

[11] The threshold ξ has been experimentally found from a current set of cases and domain solutions available, and set up to 0.96 (see (Verdaguer, 1989; Verdaguer et al., 1992)).

0.89 and the second highest one is 0.88. Since the similarity degrees of the nodes are close their situations are close and we expect that their plans only differ on one or few actions. Moreover, regarding medical diagnosis, it is interesting to note that uncertainty and incompleteness induce physicians to prefer performing additional tests rather than to forget testing dangerous diseases (conservative bias). So when the two highest similarity degree are close, we continue problem solving with a short plan (composed by one or few actions)[12]. But it makes no sense to continue problem solving if the highest similarity degree is 0.99 and the second best one is 0.11. When the similarity degree differs so much, we expect that the plans of the corresponding nodes are also very different. To continue problem solving by adapting the plan of the node with the second highest similarity degree will mean to execute a long, very different plan. Besides, a lesser similarity degree means that our reliance on the resulting adapted plan with which we continue problem solving is low. We do not expect to find significant evidences for diagnosing the patient by executing the adapted plan.

In general, among retrieved nodes we can distinguish two sets N_{exec} and N_{rest} of leaf nodes (final plans and goal states). On the one hand, the nodes N_{exec} are those whose plans have been already executed. Consistently the plans of N_{exec} do not provide information to build a new plan to continue solving the current problem. On the other hand, the nodes N_{rest} are those just retrieved according to the last information available and whose plans have not been executed. BOLERO has to decide whether or not to continue problem solving with a plan of a node in N_{rest}. If the similarity degree of nodes in N_{exec} to the current case is rather higher than any of the nodes in N_{rest}, it is reasonable to think that, if we choose a plan of a node in N_{rest} to continue, we will increase plan length unnecessarily.

Recall that the similarity degree is computed by applying the matching function m_s to the current problem solving state and the generalized situation υ stored in a node (see section 4.2.1). Particularly, when applying the matching function to leaf nodes, we talk about the similarity degree of a generalized goal state υ and the current state s_a. The key issue of detecting dissimilarities relies on comparing the similarity degree of the generalized goal states that index executed plans to the similarity degree of the generalized goal states that index plans not yet executed. For this reason we have defined a threshold[13] ψ. If υ is a generalized goal state that indexes an executed plan, υ' a generalized goal state that indexes a plan not yet used in the current case, and it is satisfied that

$$m_s(s_a, \upsilon) - m_s(s_a, \upsilon') \geq \psi$$

BOLERO considers whether the cases in memory with non-used plans are sufficiently dissimilar to the reused plans. If the non-used plans will not enlighten the current problem solving process BOLERO stops problem solving. The current state is considered the goal state of the current problem.

6 Application to pneumonia diagnosis

We have applied BOLERO to learning and generating plans to diagnose the agent causing a pneumonia in patients that acquired the illness outside the hospital environment. Pneumoniae are diseases that frequently need an urgent treatment (Verdaguer, 1989). An urgent treatment means that the physician has to diagnose the agent that causes the infection taking advantage of the known information about the patient, and before knowing the results of some lab tests. Although it is possible to realize complementary physical examinations, the physician usually do not know the etiology[14] of the illness. Incomplete and uncertain information are characteristics of the pneumonia diagnoses and they persist along all the therapeutic process. Interleaving plan elaboration and execution of plans is especially important in this application in which the incomplete information problem is becoming more acute by the necessity of making decisions and giving advice that have

[12]Note also that the monotonicity assumption about the effect of actions allows us to proceed in this way.

[13]ψ depends on the number of cases in memory and is defined in the unit interval. In (López, 1993) there is an ablation study on the effects of this parameter.

[14]Part of Medicine that studies the agents causing illness.

consequences on the patient. We need to make decisions and the experience of the physician in solving past cases plays a fundamental role in the patient diagnosis.

To develop the application we used 79 cases provided by a teacher that correspond to real cases from four hospitals (Verdaguer, 1989). The selection of cases has been done by the teacher, and follows a stratified random procedure based on the diagnostic groups of the WHO (Verdaguer et al., 1992). We used 460 attributes to represent the domain of pneumonia. Each case has, on average, 88.23 attributes, 33% of which are attributes with unknown value. Unknown values (a.k.a. incomplete information) are thus an essential feature of this domain. To execute the plans generated by BOLERO, we used PNEUMON-IA* (Verdaguer, 1989; López, 1993), a rule-based system able to interpret BOLERO's plans. PNEUMONIA* acts as the object-level of BOLERO-RBS, while BOLERO acts as the meta-level. The evaluation process uses 79 evaluation standards, one for each case, based on the teacher advice plus the consensus of five other expert opinions.

BOLERO has been implemented on a SUN, using CLOS (Common Lisp Object System). We followed an experimental evaluation method to estimate the system performance. An experiment consists of using BOLERO to learn plans from $n-1$ cases among the n available cases of an evaluation sequence $S = [C^1, C^2, \ldots, C^n]$, and then measuring the identification and prediction of the system by using cases 1 and n respectively (*leave one out* method). *Identification* tell us if the system is able to successfully solve a problem identical to a past problem, i.e. a case already learned. *Prediction* indicates the correctness and precision of the system when solving a problem previously unseen by the system.

As in any learning system, BOLERO is sensitive to the quality of cases. If we train BOLERO with typical cases and then we test it with atypical cases, results will be unsatisfactory. On the contrary, false results can be obtained if we do not select accurately training and test cases. To guarantee that the results do not depend on the order of cases we have randomly generated 10 different evaluation sequences and the results shown in this paper report the average of the 10 sequences. To complete the experimental evaluation we have also performed an ablation (ironing or lesion studies (Cohen & Howe, 1988; Kibler & Langley, 1988)) of the system in which it is possible to see the impact of the different components of the system and design decisions on the system efficiency and results. The following section shows the results obtained when testing 10 evaluation sequences on BOLERO according to the incorporation, retrieval, adaptation and evaluation methods. Moreover, the strategic relevance of all facts is initially set to 0.5. In the following, we provide the results of the ablation process in which other methods are incorporated (learning from experience, changing the strategic relevance or facts, etc.).

6.1 Results on identification

Cases identical to a past experience always are solved correctly by BOLERO, which means that the degree of success for the case is 100%, the degree of focus is 100%, and the degree of disorder is 0%. So, we say that BOLERO solves a case identical to another in its memory in the same way that the teacher does. It is important to note that the number of cases in memory does not affect the identification of the system.

6.2 Results on prediction

Regarding prediction, BOLERO, as we expected, is not so good as in identification, but the results are quite satisfactory: the degree of success approaches 98,30% as shown in figure 8. At the beginning, BOLERO has no knowledge about plans, but the degree of success increases rapidly because it learns quickly most of the actions. We have observed that a plan in our application is quite large, so after learning 8 cases, BOLERO knows 76,67% of all the actions of the diagnostic task. Finally, the degree of success increases up to 98,30%.

The degree of success achieved is acceptable if we analyze the features of our teacher. We have compared the diagnoses provided by our teacher with the diagnoses provided by five experts, and we observed that our teacher presents more etiologies than the others. Moreover, in our

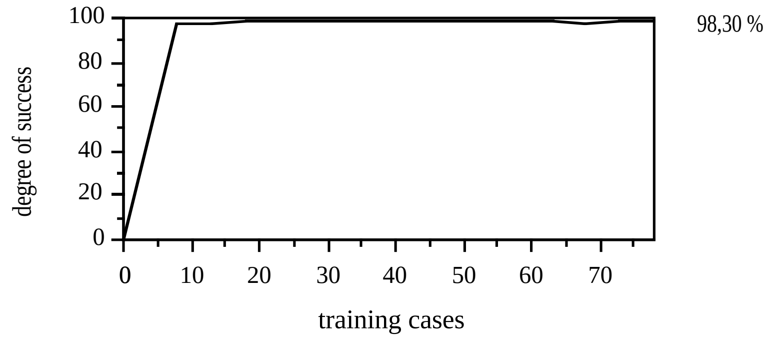


Figure 8: Degree of success in prediction.

comparison among the diagnoses of the different experts we have detected missing etiologies, that is, etiologies that are provided by every expert except one. Missing etiologies can be related to the degree of success: if an expert has a missing etiology, then his degree of success is less than 100%. Performing a deeper analysis we found that every expert presents at least one missing etiology. Summing up, the fact that our teacher is the expert that shows the greater number of etiologies and that it is a normal thing to detect missing etiologies among experts, the degree of success 98,30% achieved by BOLERO can be considered satisfactory.

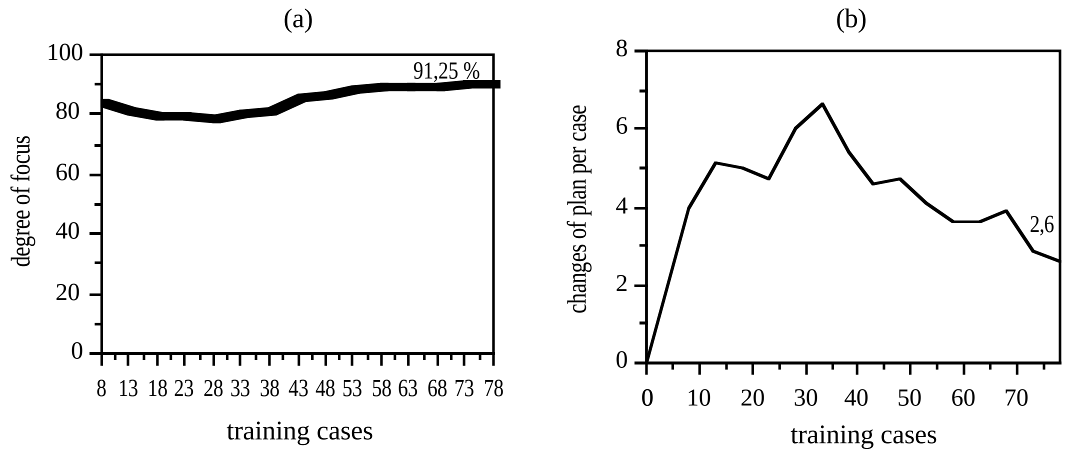


Figure 9: Results on prediction. (a) Degree of focus. (b) Unsteadiness.

The degree of focus on prediction is shown in figure 9a[15]. It is possible to observe that the number of cases in memory does not affect negatively the degree of focus, as occurs on identification. In fact, we have observed that the number of times BOLERO changes the plan when solving a problem (what we call degree of *unsteadiness*) decreases as the number of cases increases (see figure 9b). At the beginning (25 first cases), BOLERO has an unsteadiness of 4-5 changes of plans per cases (*cppc*). This is caused by the fact that BOLERO has few cases in memory and, as a consequence, there are few plans to choose when doing problem solving. Afterwards, when there are about 25-40 cases in memory, the degree of unsteadiness is quite high: up to 7 cppc. The number of plans in memory has been increased and so BOLERO has more chances, more combinations to solve a problem. From this point on, when BOLERO learns more

[15]Initial data (data between 0 and 8) are not printable. If we assume that BOLERO does not have any knowledge about plans, then it does not plan. The degree of focus cannot be applied to a null plan.

cases, unsteadiness is reduced up to 2,6 cppc. The number of cases has been increased, and also the number of alternatives, but the possibility of finding a similar case in memory is also higher. Lower unsteadiness then means that BOLERO finds quickly the most similar plan in memory and that with few changes it is possible to elaborate the complete plan. Note that we never expected an unsteadiness of 1. This means that the information about a problem is complete and precise, and this does not happen in the domain of medical diagnosis. If the information is complete and precise, there is no need to develop an architecture for interleaving plan elaboration and plan execution. So, a degree of unsteadiness of 2,6 cppc is quite good.

Regarding the degree of disorder when more cases are added in memory, BOLERO tends to solve problems by building plans where the order of actions is slightly different from the order of the actions that follows a teacher's plan. In spite of that, BOLERO increasingly improves the order of actions (as is assessed by the decrease of disorder degree) during the sequence of experiments. We can summarize here that the case-based methods implemented allow to achieve similar results (and even better, as the case of the degree of disorder) to the ones showed by the human expert in diagnosis.

6.3 Execution time

The time consumed by BOLERO when learning a case increases with the number of cases (see figure 10a). This increase is evident since with more cases in memory BOLERO has more nodes with which to compare a new case, more data to perform generalizations, etc. Nevertheless, the final time is less than the time required to acquire from a human expert the same strategic knowledge needed for planning. In fact, we have a version of the rule-based system PNEUMON-IA*, called PNEUMON-IA, that contains the strategic knowledge coded in metarules. Whereas PNEUMON-IA* has a unique plan composed by all possible actions (*[gather-information, bact-atyp, pneumococ, enterobact, staphyloc, tbs, hemoph, branha, anaerobis, streptococ, legionella, meningo, pseudo, nocardia, criptococ, myco, chlamydia, q-fever, viruses, viruses-bact, citomegalovirus, aspergillus, pneumocistii]*) to solve any case, PNEUMON-IA has 64 metarules that generate dynamically plans of actions for each case. Using BOLERO, PNEUMON-IA* behavior has been improved, in such a way that its final behavior is closer to the one shown by PNEUMON-IA. The expert spent approximately 14 days (8 hours a day) in developing the metarules of PNEUMON-IA. So we get a great improvement using BOLERO to improve PNEUMON-IA* performance[16].

Regarding the time consumed in solving a case, time increases linearly with the number of cases, with an slope of 3.5 (approx.), as it is shown in figure 10a. In spite of that, the response time of BOLERO is acceptable in an interactive environment. Nevertheless, the increase of the execution time with respect to cases in memory, has lead us to think on developing a mechanism to decide the convenience of incorporating a case in memory, instead of incorporating all cases, as we are currently doing.

6.4 Plan memory breadth and depth

The depth of the hierarchy of plans in memory is 12.9367 nodes in average (variance 4.6179) and the breadth is 1.13 children per node (variance 0.5263). Since the total number of actions in our domain is 24, we can conclude that the plans are large (around 50% of the actions). We have observed that there are few branches and most of them are clustered at the root. From these few branches nodes have only one succeeding node up to certain depth. Branching appears again on the lowest levels of the hierarchy. When we are closer to a leaf node plans seem to be strongly differentiated, that is, we have more information and so less indecision in the plans. In some way, we can talk about a few number of prototypical subplans. Any plan generated is built by using a prototype and enlarged with some specific actions, until a goal state is achieved.

[16]Note that cases needed for learning should be also required to build PNEUMON-IA and verify its correct behavior. So the comparison we make here is valid and illustrates the potential benefit of using learning techniques in the acquisition of plans.

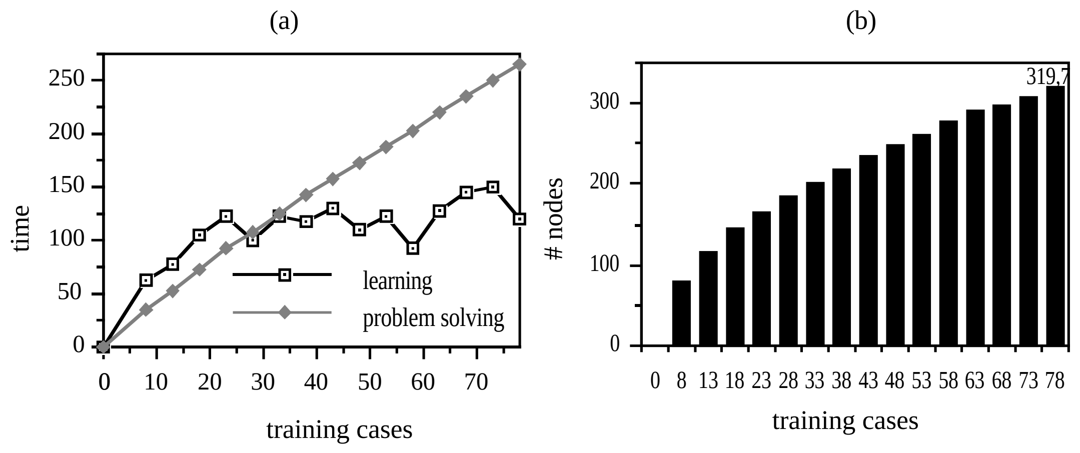


Figure 10: Results on prediction. (a) Execution time spent in learning and problem solving. (b) Number of nodes generated.

The number of nodes in memory increases with regard to the number of training cases, as shown in figure 10b. However it is interesting to note that we do not have an enormous number of plans as a consequence of the presence of those prototypical subplans. The number of plans tends to converge as we can see in the evolution of the learning curve of figure 10b.

6.5 Ablation

We have completed the experimental evaluation of BOLERO by modifying some features of the system and then studying their impact on the results. We have performed several variations. The more representative results are shown in this paper (for the rest see (López, 1993)).

6.5.1 Multiexpert versus teacher-only evaluation

To evaluate the effects of the evaluation procedure we have replaced the evaluation standard (ES) obtained from the consensus of several experts as explained in section 4.3 (multiexpert evaluation) by a ES based exclusively on the teacher's advice. Our expectation was that multiexpert evaluation favors larger plans, since taking into account the opinion of different experts we are accepting more possibilities (i.e. actions) in a plan. Our expectations have been confirmed when the results of the system remain unchanged except for the degree of focus on prediction that decreases slightly (see table 4b). The fact that actions not considered by the teacher are accepted introduces a conservative bias, although it goes against the parsimony criterion.

6.5.2 Equal versus different strategic relevance of facts

The next alteration made in BOLERO consists of assigning a different strategic relevance degree to each fact (instead of an initial relevance degree set to 0.5 for all facts). Relevance setting has been performed by means of the hierarchy of facts expressed in the domain network. For example, the strategic relevance degree of facts whose identifier belongs to the class *semiology* are set to 0.2, while the strategic relevance degree of facts belonging to *lab-data* are set to 0.9. The importance of the classes has been established by the teacher following the reliance he has in the facts of each class when elaborating his diagnostic procedure. For instance, since semiological data are quite subjective they can be interpreted in several ways. Consistently, the facts related to the class *semiology* have a low strategic relevance. Laboratory data, on the contrary, are objective

Table 4: Results on prediction: degree of focus.

% cases	(a)	(b)	(c)	(d)
10.36	85.22	82.86	85.22	85.22
16.67	82.23	80.01	81.73	82.76
23.08	80.24	78.02	81.85	83.58
29.49	80.30	78.20	83.99	83.11
35.90	79.88	78.21	81.71	83.82
42.31	80.93	79.26	83.43	85.76
48.72	82.43	80.76	87.43	87.93
51.13	87.01	85.35	88.74	88.35
61.54	97.92	86.25	90.83	88.75
67.95	89.17	87.5	92.08	88.75
74.36	90.00	88.33	91.67	92.08
80.77	90.42	88.75	91.67	93.75
87.18	90.42	88.75	93.33	95.00
93.59	91.25	89.21	94.17	95.00
100	91.25	89.58	94.17	95.00

(a) Multiexpert, learning by experience and equal strategic relevance for each of the 460 facts (full set). (b) Evaluation standard based exclusively on the teacher's opinion. (c) Assigning to each fact a specific strategic relevance. (d) Reducing the set of facts from 460 to 19.

and trustworthy when making a diagnosis. For this reason, the teacher assigns to lab-data facts a higher degree of strategic relevance than to semiology facts.

As we expected, the performance of the system has been improved. The degree of success remains 100% but the degree of focus increases from 91,25% up to 94,17% (see table 4c). These results show that the strategic relevance of facts is a good instrument to index cases in the system.

6.5.3 Learning from a teacher versus learning from experience

The results shown in sections 6.1 and 6.2 refer to BOLERO when using the learning from a teacher method. Now we take into account learning from experience. Initially the strategic relevance degree is the same for every fact (0.5). The strategic relevance degree is then refined using the method of section 4.4.2. We expected to eventually obtain the strategic relevance degree of facts quite differentiated in order to achieve the same results that the ones obtained when the teacher assigns a different strategic relevance degree to each fact.

The results show that, although the degree of focus does not reach the score of 94,17%, it follows an increasing learning curve up to 91,25% (see table 5). However, a new sign appears: for the first time, the degree of success increases up to 98,86%. This is only a slight difference, but it is a sign that tells us that learning the strategic relevance degree of facts we can achieve better results that obtaining the strategic relevance degree of facts from the teacher. Besides, we should take into account that when learning the strategic relevance degree of facts we free the expert of providing them in the knowledge acquisition phase of the system development.

6.5.4 The effects of the conservative bias

There is a conservative bias in the system that comes from the definition of the degree of success (see section 4.3). In the lesion study that follows, we have avoided such bias. The results show an increase in the degree of success in prediction: 100% when the system has been trained with 8 cases. The rest of the measures remains the same.

In such configuration we observe that the system learns to build plans whose actions are*consolidated*. That is, an action is consolidated if its execution produces a positive step towards the solution of the problem. For example, if the action *pneumococcus* leads to produce the fact *(pneumonia-by-pneumococcus, very-possible)*, we say that the action has been consolidated.

Table 5: Degree of success and degree of focus on prediction when learning from a teacher (a) and when learning from experience (b).

casos	degree of success		degree of focus	
	(a)	(b)	(a)	(b)
10.36	91.19	97.19	85.22	82.86
16.67	97.75	97.75	82.23	80.01
23.08	98.89	98.89	80.24	78.02
29.49	98.33	98.33	80.30	79.86
35.90	98.33	98.33	79.88	78.21
42.31	98.33	98.33	80.93	80.10
48.72	98.33	98.33	82.43	83.68
51.13	98.33	98.33	87.01	85.76
61.54	98.33	98.33	97.92	86.67
67.95	98.33	98.33	89.17	87.92
74.36	98.33	98.33	90.00	88.75
80.77	98.33	98.33	90.42	89.17
87.18	97.75	98.30	90.42	89.58
93.59	98.30	98.86	91.25	91.67
100	98.30	98.86	91.25	91.25

However if the execution of *pneumococcus* produces the fact *(pneumonia-by-pneumococcus, almost-impossible)*, we say that, in some sense, the action fails. In a robot navigation environment, it is essential to learn to plan consolidated actions while avoiding failed actions. This is not fundamental in medical diagnosis, where it is important to consider a dangerous disease in order to check it is really unlikely. The results when modifying the conservative bias show that BOLERO can be applied to domains different from medical diagnosis.

However we should not forget that BOLERO has been designed to be useful in medical diagnosis, as the bias introduced in the system confirm. For example, therapy applications are not well suited to BOLERO, mainly due to the monotonicity assumption on situations. Applications where BOLERO could be useful are fault diagnosis: finding which component of a circuit does not work, which part of a car is causing problems, etc. Currently, we are applying BOLERO to the diagnosis of connective tissue diseases and inflammatory arthropathies (rheumatology (Belmonte, 1990), for more details see (López, 1993)). The development of two different medical applications, together with the fact that BOLERO could be useful for diagnostic task applications, allows us to hypothesize about the utility of BOLERO for several domains where the monotonicity hypothesis holds.

6.5.5 Strategic set of facts versus full set of facts

The last lesion study of the system consists of evaluating the impact of the vocabulary used by BOLERO, that is, in the quantity and quality of the domain knowledge. It is well known that an important fact that affects learning is the presence of irrelevant facts (Kibler & Langley, 1988). Irrelevant facts are those facts of cases that are not relevant for the diagnosis of the patient. For example, the name of the patient is an irrelevant fact, while the X-ray data are relevant. The presence of irrelevant facts in the diagnostic procedure impairs the learning capabilities of the system and diminishes the performance of the system, as the results of our lesion study demonstrates.

Our teacher has selected among all the vocabulary (460 facts) the set of 19 facts strategically relevant. Now cases are represented by a lower number of facts and so are the nodes. Given a situation, few nodes are now recovered from memory, but they are the relevant ones. In this experiment the results are the following: with only 23,08% of the training cases, BOLERO achieves a degree of success of 100%, and the degree of focus reaches 95,00% (see table 4d).

As the number of facts is reduced, the number of indexes is also diminished, and the execution time is significantly reduced. Learning time is about 3,5" (instead of 119,05" when working with the complete vocabulary) and problem solving time is 28,25" (instead of 265,77"). The important thing is that the time curve stops increasing and it stabilizes around 18 training cases (see figure 11b). Regarding overall time we should take into account the time needed by the teacher to select the strategic facts from the overall set of facts. The full set of facts is the vocabulary from which we start, that is, the facts represented in the cases collected from the hospitals. This experiment shows that reducing the vocabulary will be beneficial since results are improved. But reducing the set of facts has a cost on the expert knowledge acquisition time. These results encourage us to develop in a future a learning method to automatically reduce the amount of facts. We think that in case of having a set of cases larger than the current one the method to learn the strategic relevance degree of facts will reduce automatically the number of facts by lowering down the strategic relevance degree of irrelevant facts and finally erase the irrelevant indexes of nodes. In fact, we have already tried to implement this method but we would need more than the 79 hospital cases available to achieve significant results in the pneumonia diagnosis. Since changes are performed smoothly decreasing the strategic relevance of a fact to 0 requires a lot of training cases.

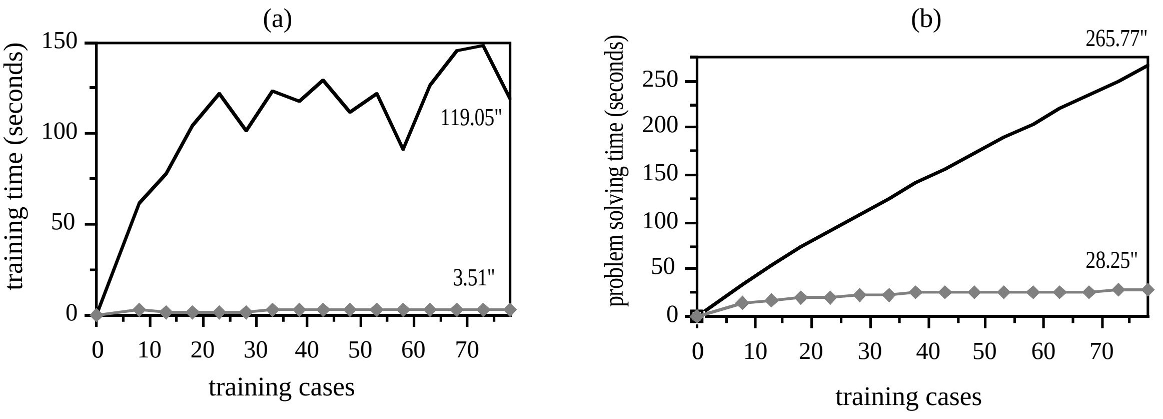


Figure 11: Time consumed in training (a) and problem solving (b) when reducing the vocabulary. Black line reflect the results for the full set of facts; grey line the ones for the strategically relevant facts.

7 Related work

There are three main research topics we related in BOLERO: case-based reasoning, case-based planning, and planning.

7.1 Case-based reasoning

Case-based reasoning is a technique often used to develop KBS because of its advantages in knowledge acquisition. However, each system uses a particular case representation adequate to the task at hand, and each system uses specific retrieval, adaptation and learning methods.

In order to store cases most of the case-based systems go beyond representing generalized cases in memory and represent individual, concrete cases (JULIA (Kolodner, 1987)). In BOLERO we have represented generalizations of situations, specific plans (individual), and the solutions to cases in leaf nodes. Regarding the generalization method, any case-based system presents its own procedure. In this way, for example, it is interesting to note that REFINER (Sharma & Sleeman, 1988) generalize the facts by using a semantic net, in the same way as BOLERO (i.e., the facts *orange* and *apple* are generalized in the fact *fruit*).

The similarity metric used in the retrieval process is crucial, but there are few formal studies performed on this topic. Bonissone and Ayud (1992) propose the use of linguistic values associated with semantic values in order to index and retrieve cases from memory. In HYPO (Ashley & Rissland, 1988) a credit assignment process is employed. In BOLERO, we have used numerical values to compare cases because either the use of linguistic labels or a credit assignment process require an additional effort in knowledge acquisition as well as in the treatment. In spite of the fact that similarity functions are often defined ad hoc in each system, it is important to observe that, as we have done in BOLERO, most of the authors use the strategic relevance of facts to evaluate the similarity among cases (Bonissone, Bloom (Aha, 1989), CABOT (Callan et al., 1991), Cain (Cain et al., 1991)).

Regarding the learning from experience methods, all systems have the same problem: to determine which cases to store. Lenhert (Lenhert, 1987), for example, has concluded that the increment of cases in the system can worsen the system performance rather than improve it. Utility measures has been studied better in other machine learning methods (as in PRODIGY-EBL (Minton, 1988)) than in CBR.

We want to stress that in BOLERO we distinguish a learning phase different from learning from experience, that we call learning from a teacher. In BOLERO the acquisition of precedents is explicitly distinguished in a learning phase in which it learns from a teacher.

With respect to the learning from experience mechanism, the method for updating strategic relevance degree of facts implemented in BOLERO is related to the modification process of the relevance of indexes in PROTOS (Porter et al., 1990). PROTOS is quite different from BOLERO. Firstly, instead of learning plans, its main objective is to learn concepts from cases provided by a teacher and a different organization of memory is required to deal with such different kind of problems. PROTOS organization of cases is richer than the one developed in BOLERO, since PROTOS has a great variety of links that are used in different situations: to retrieve past cases, to determine feature exclusion, to differentiate near-miss cases, etc. Indexes are established when learning a new case and each index has an importance degree attached that reflects the strength between a feature and a case or exemplar. This importance degree is adapted successively when more cases are added to memory and by establishing a dialogue with the expert. This is an important difference from BOLERO that learns the strategic relevance automatically. In general, PROTOS strongly depends on a dialogue with the teacher: the teacher decides if a classification is correct or not, provides the explanations, etc. BOLERO only needs the expert in providing the training cases and the validity of the solutions.

The results of BOLERO show it is possible to improve a rule-based system by acquiring planning knowledge from cases. The cost of hours of work that a knowledge engineer would required to hand-code rules to implement such planning knowledge is saved. CASEY (Koton, 1988) was one of the first systems that shows this improvement by integrating a case-based system with a causal reasoning system in order to diagnose heart failures. The causal reasoning system provides robustness to the overall system meanwhile cases enlighten the solution of the current case by referring to previous cases and thus improving the efficient of problem solving. It is interesting to note that both CASEY and PROTOS are based on the elaboration of explanations when justifying missing features in cases. The spreading activation method used in BOLERO deals with missing features tacitly in the retrieval process. Features of a new case activate facts in the domain network that are used as indexes. A missing feature can be included in the matching function by a fact more general than the missing feature. We think that this approach simplifies BOLERO's retrieval process as compared to those of CASEY or PROTOS. Another interesting issue is that BOLERO learns protocols, that is, how to diagnose a patient, whereas CASEY and PROTOS learn diagnoses. In fact, they deal with learning in different kinds of tasks. In this sense, the task of medical diagnosis has required the development in BOLERO of a method to learn and recognize goal states.

7.2 Case-based planning

There is a big variety of case-based planners but it is interesting to note that most of them work on a two-phase scenario: first a plan is elaborated, and then this plan is repaired if it has failed (NEGOTIATOR (Kolodner & Simpson Jr., 1986), CHEF (Hammond, 1989)). In contrast to them, BOLERO is able to replan dynamically whenever the current plan is not adequate to the current situation. This adaptive capability is a consequence of integrating BOLERO with a rule-based system in a meta-level architecture.

Prodigy/Analogy brings efficiency to NoLimit, in the same way that BOLERO does to RBS. Prodigy/Analogy interacts with NoLimit, a nonlinear planner ascribed to the PRODIGY architecture (Veloso, 1992), to solve problems. Plan representation in Prodigy/Analogy is richer than the representation chosen in BOLERO. Prodigy/Analogy uses a tree to represent plans in which it is shown the nonlinearity of plans and it keeps track of the conditions that support past plans. Plans are retrieved and replayed in new cases. Both Prodigy/Analogy and BOLERO can solve a problem with multiple cases but the approaches are different. Prodigy/Analogy can use different cases while subgoaling during plan elaboration. BOLERO uses more than one case as a consequence of adapting the reasoning process by re-planning and can generate new plans by means of replanning and the reuse of parts of retrieved plans. Another important issue about Prodigy/Analogy is that it cannot learn from a teacher, as BOLERO does. When Prodigy/Analogy has no case in the memory, it expects that NoLimit solves the cases by using alternative strategies (control rules, or asking questions). Cases solved by NoLimit constitute the precedent cases of Prodigy/Analogy. The main difference among the cases provided by the teacher in BOLERO and the ones provided by NoLimit in Prodigy/Analogy, is based on the fact that the cases of NoLimit include information about the available operators in a decision steps, which operator has been selected, and other kind of information that a problem solver can provide, but that is difficult to acquire from a human expert. The different kind of domains for which the systems have been designed (BOLERO for open-goal planning in medical diagnosis and Prodigy/Analogy for classical planning) have been decisive in the approach followed in each system to acquire cases. In this sense, BOLERO planning is based on actions while Prodigy/Analogy is based on operators.

Other interesting case-based planning systems related to BOLERO are TRUCKER and RUNNER (Hammond, 1988). Planning in TRUCKER and RUNNER means to re-use previous plans. When a goal cannot be achieved, it is suspended and no other alternative method is used. When executing other plans, new information can recall the suspended goals. As a consequence, a new plan is generated in which current goals and suspended goals are achieved. This re-planning capability is similar to the one showed in BOLERO: when the information changes, there is a chance for re-planning. The organization of cases in memory in TRUCKER and RUNNER is based on the opportunities to recall suspended goals, that is in situations in which suspended goals could be achieved. But situations in BOLERO are collections of all the information known about a case (patient) instead of particular information about a particular action or goal. It is possible that in this sense TRUCKER and RUNNER have a better retrieval time than BOLERO.

7.3 Planning

Among some planning system we distinguish four interesting ones from the point of view of our research: Theo-Agent, PROTÉGÉ, ASK and PRODIGY/EBL. Theo-Agent (Mitchell, 1990) is a reactive planner that learns from experience. Our system differs from Theo-Agent in that BOLERO is able to react to the environment by adapting the reasoning process to changes, i.e. by replanning. As a consequence, whereas Theo-Agent learn instantaneous reactions to particular situations, our system learns complete plans. Another drawback typical of reactive planners that our system does not have is that Theo-Agent does not become necessarily more correct given that stimulus-response rules does not reflect the effects of the actions.

PROTÉGÉ (Tu et al., 1992) is a system that provides knowledge acquisition tools for the development of knowledge based systems applied to the medical domain. In such domains, we concur with the authors that plans must vary when the information gathered from the environment changes. PROTÉGÉ has been applied to a special kind of problem where changes are expected: therapy (chemotherapy and radiation therapy). Changes come not from knowing new information, but because of nonmonotonicity: some values of facts change as the effect of some previous actions. For example, the value of fact *fever* can change from 39 centigrade degree to 37 as an effect of the therapy. In BOLERO, however, changes occur due to the incomplete and uncertain information known about a patient while being diagnosed, in a given session, and monotonicity is assumed. Moreover, the knowledge acquisition tools provided in PROTÉGÉ-II are based on a fixed sequence of tasks, whereas the purpose of BOLERO is precisely to organize the task (goals to validate) dynamically according to the most recent information available.

ASK (Gruber, 1989) is a system designed to learn strategic knowledge to plan. The experimental evaluation of ASK has demonstrated that it is not quite efficient in learning from scratch. BOLERO however, is able to learn from scratch by means of the learning by observation procedure introduced in this paper. Moreover, BOLERO can learn either from success or from failure while ASK only learns from failure.

However, the learning from experience method developed in BOLERO leads to an accumulation of cases in memory that punishes the efficiency of the system. In this sense we think that we need to incorporate utility measures like the procedures given in PRODIGY/EBL (Minton, 1988). The utility measures of PRODIGY/EBL are based on the applicability of the control rules learned. The utility of cases could be measured in a similar way, but it is no so simple: cases not often used can keep information about singular experiences (i.e. especial critical patients) that should be taken into account.

8 Conclusions and future work

The main contributions of our research work, materialized in the experiments performed with BOLERO, can be summarized in five points. First, we have developed a case-based planning method for knowledge acquisition that relieves the development of complex, real world applications. With such a method, it is possible to learn from cases provided either by a teacher or by the system's own experience. Advantages of BOLERO are the saved engineer's time in knowledge acquisition, and the automatic detection and correction of interactions among different parts of knowledge (i.e. tuning work). With BOLERO we have shown how case-based planning minimize the scaling up problem that some machine learning techniques may present.

Case-based planning is not only a useful method for acquiring plans but also for generating plans. In a real world application the quality and amount of data manipulated makes planning a complex task. In this sense, the definition of the strategic relevance of facts has been fundamental. Working with a real work application, however, introduces a constraint: the availability of cases. That leads us to assume that a set of significant cases that allows the learning are available.

Second, the case-based planner we built is able to retrieve plans from memory according to the information available anytime. This feature is quite important if we take into account that in many real world situations known information is incomplete and uncertain. The approach taken to represent cases has been relevant to develop the retrieval method that allows such behavior. Cases are represented by decision steps, in such a way that parts of past plans can be used to solve a new problem. As a consequence, it is possible to generate new plans not previously in memory. The analysis done about unsteadiness shows that, in spite of the uncertain and incomplete data, the automatic storing mechanism (with the corresponding generalization method) and the retrieval mechanism are adequate to the task performed by the system.

Third, we have implemented a way to improve the problem solving efficiency of an existing rule-based system by integrating it with the case-based planner in a meta-level architecture. In the meta-level architecture the solution of a problem is achieved by interleaving plan elaboration and plan execution, in such a way that the reasoning process is adapted to changes in the environment. The third main contribution is then the meta-level architecture with which to achieve an adaptive planning behavior in problem solving that improves over time.

And fourth, we have developed a method to learn to recognize goal states to deal with open-goal planning. This is an innovative issue that arises while learning to plan in medical diagnosis. The results also manifest the presence of a conservative bias that specially characterizes medical diagnosis, and that limits the learning capabilities of the system in domains different than diagnosis.

We know that our system has some weak points, as for example the parameters that we have determined experimentally, the monotonicity assumption and the conservative bias, among other. But we believe that the experimental results achieved with BOLERO are meaningful for the development of real, complex applications.

As future work we have several works to improve BOLERO and some new research directions out of the scope of BOLERO. In order to improve BOLERO firstly we look for a definition of a utility measure for cases. Along the the experimental evaluation of BOLERO we have seen the necessity of defining a utility measure in such a way that BOLERO only incorporates in memory cases that improve the knowledge of the system (i.e. with plans significantly different). Secondly we could study a *chunking* mechanism to reduce the number of memory nodes (reducing also the retrieval time). Node reduction can be performed through the identification of sequences of nodes with a unique successor, i.e. without branching. Such sequences of nodes can be compacted or chunked in a single node. Thirdly we think about a mechanism for distinguishing relevant from non-relevant facts. The experience in BOLERO has shown that incrementing the number of facts worsens the response time of the system as well as the correctness and accuracy of plans. Although it is possible to know beforehand relevant from irrelevant facts, this task requires an additional effort on behalf of the expert in knowledge acquisition time. Therefore, it is interesting to study new mechanisms that learn the strategic relevance of facts and erase irrelevant facts from memory. And finally, we are developing a method of generalizing solutions in order to reduce the cost of the storage of all possible solutions needed to learn and recognize goal states. Generalized solutions could reduce both memory and time when trying to recognize a goal state.

In the line of new research directions we plan to study a method to learn failures. A method to learn failures is different from a method that learns from failures. Learning failures allows us to retrieve previous errors in solving new problems. Together with a method to learn failures, a method to repair plans is under study.

References

Agustí, J., Esteva, F., García, P., Godó, L., , López de Màntaras, R., Puyol, J., Sierra, C., and Murgui, L., (1992). Structured Local Fuzzy Logics in MILORD. In Zadeh, L. and Kacprzyk, J., editors, *Fuzzy Logic for the Manegement of uncertainty*, Wiley-Inter Science, Wiley Professional Computing Series, pages 523–551. John Wiley & Sons, Inc.

Aha, D., (1989). Incremental Instance-Based Learning of Independent and Graded Concept Descriptions. In *Proc. IWML.*

Ashley, K. and Rissland, E., (1988). Waiting on Weighting: A Symbolic Least Commitment Approach. In *Proc. AAAI.*

Belew, R., (1992). Paradigmatic over-fitting. Personal communication.

Belmonte, M., December 1990. *RENOIR: Un Sistema Experto para la Ayuda en el Diagnóstico de Colagenosis y Artropatías Inflamatorias.* PhD thesis, Universitat Autònoma de Barcelona.

Bonissone, P. and Ayud, S., (1992). Similarity Measures for Case-Based Reasoning Systems. In *Proc. IPMU. International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 483–487. Universitat de les Illes Balears.

Bonissone, P. and Decker, K., (1986). Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off Precision and Complexity. In Kanal, L. and Lemmer, J., editors, *Uncertainty in Artificial Intelligence*, pages 217–247. Elseview Science Publishers B.V., North-Holland.

Cain, T., Pazzani, M., and Silverstein, G., (1991). Using Domain Knowledge to Influence Similarity Judgments. In *Proc. CBR Workshop.* DARPA.

Callan, J., Fawcett, T., and Rissland, E., (1991). Adaptative Case-Based Reasoning. In *Proc. CBR Workshop*, pages 179–190. DARPA.

Carbonell, J., (1991). Scaling up Knowledge Based Systems via Machine Learning. Key-note presentation in the European Working Session on Learning, 1991.

Cohen, P. and Howe, A., (1988). How Evaluation Guides AI Research. *AI Magazine*, 9(4).

Compton, P., Edwards, G., Karg, B., Lazarus, L., Malor, R., Preston, P., and Srinivasan, A., (1992). Ripple Down Rules: Turning Knowledge Acquisition into Knowledge Maintenance. *Artificial Intelligence in Medicine*, 4(6):463–475.

Cornuejols, A., (1993). Memory Limitations and Optimization of Training Sequences for Incremental Learning. In de Velde, W. V., editor, *MLNet Workshop on Learning in Autonomous Agents*.

Dubois, D. and Koning, J. L., (1991). Social Choice Axioms for Fuzzy Set Aggregation. *Fuzzy Sets and Systems*, 43(3).

Elio, R. and Scharf, P., (1990). Modeling Novice-to-Expert Shifts in Problem-Solving Strategy and Knowledge Organization. *Cognitive Science*, (14):579–639.

Gennari, J., Langley, P., and Fisher, D., (1990). Models of Incremental Concept Formation. In Carbonell, J., editor, *Machine Learning. Paradigms and Methods*, pages 11–61. MIT/Elsevier.

Gruber, T., (1989). Automated Knowledge Acquisition for Strategic Knowledge. *Machine Learning*, 4:293–336.

Hammond, K., (1988). Opportunistic Memory: Storing and Recalling Suspended Goals. In *Proc. CBR Workshop*. DARPA.

Hammond, K., (1989). *Case-Based Planning. Viewing Planning as a Memory Task*, volume 1 of *Perspectives in Artificial Intelligence*. Academic Press, Inc.

Kibler, D. and Langley, P., October 1988. Machine Learning as an Experimental Science. In Sleeman, D., editor, *Proc. EWSL*, pages 81–92, Glasgow, UK. Turing Institute, Pitman.

Kolodner, J. and Simpson Jr., R., (1986). Problem Solving and Dynamic Memory. In Kolodner, J. L. and Riesbeck, C., editors, *Experience, Memory and Reasoning*, chapter 6. Lawrence Erlbaum Associates, Publishers, Hillsdale, N.J.

Kolodner, J., (1987). Extending Problem Solver Capabilities Through Case-based Inference. In *Proc. Fourth IWML*.

Koton, P., (1988). Reasoning About Evidence in Causal Explanation. In *Proc. AAAI*.

Lenhert, W., october 1987. Case-based Reasoning as a Paradigm for Heuristic Search. Technical Report 107, Department of Computer and Information Science, Univ. Massachussets.

López, B. and Plaza, E., November 1989. Aprendizaje de Conocimientos de Control para Sistemes Expertos. In *Actas III Reunion Técnica de la Asociació Española para la Inteligencia Artificial*, Madrid,.

López, B., (1991). CONKRET: A Control Knowledge Refinement Tool. In Ayel, M. and Laurent, J., editors, *Validation, Verification and Test of Knowledge-Based Systems*, chapter 13, pages 191–203. John Wiley & Sons.

López, B., (1993). *Aprenentatge de plans per a sistemes experts.* PhD thesis, Universitat Politècnica de Catalunya, Facultat d'Informàtica de Barcelona.

Maes, P., (1988). Issues in Computational Reflection. In Maes, P. and Nardi, D., editors, *Meta-level Architectures and Reflection*, pages 21–35. Elsevier Science Pub. B. V.

Minton, S., (1988). *Learning Effective Search Control Knowledge: An Explanation- Based Approach.* PhD thesis, Computer Science Department, Carnegie Mellon University. Technical Report CMU–CS–88–133.

Mitchell, T., (1990). Becoming Increasingly Reactive. In *Proc. AAAI*, pages 1051–1058. THEO.

Pazzani, M. and Sarrett, W., (1992). A Framework for Average Case Analysis of Conjuntctive Learning Algorithms. *Machine Learning*, 9(4):349–372.

Pearl, J., (1988). Evidential Reasoning Under Uncertainty. In Shrobe, H. and AAAI, , editors, *Exploring Artificial Intelligence. Survey Talks from the National Conference on Artificial Intelligence*, chapter 10, pages 381–418. Morgan Kaufmann Publishers, Inc.

Piera, N., Desroches, P., and Aguilar-Martin, J., (1989). LAMDA: An Incremental Conceptual

Clustering Method. Technical Report 89420, Laboratoire d'Automatique et d'Analyse des Systèmes (LAAS), 7, Avenue du Colonel Roche, 31077 Toulouse Cedex., France.

Porter, B., Bareiss, R., and Holte, R., September 1990. Concept Learning and Heuristic Classification in Weak-Theory Domains. *Artifical Intelligence*, 45(1–2):229–263.

Redmond, M., (1990). Distributed Cases for Case-Based Reasoning; Facilitating Use of Multiple Cases. In *Proc. AAAI 90*, volume 1, pages 304–309, Boston, Massachussets. AAAI Press/ The MIT Press.

Sharma, S. and Sleeman, D., (1988). REFINER: A Case-Based Differential Diagnosis Aide for Knowledge Acquisition and Knowledge Refinement. In Sleeman, D., editor, *Proc. EWSL*, pages 201–210.

Tate, A., (1990). A Review of Planning Techniques. In Allen, J. and Tate, A., editors, *Readings in Planning*, pages 26–49. Morgan Kaufmann Publishers, Inc.

Tu, S., Shahar, Y., Dawes, J., Winkles, J., Puerta, A., and Musen, M., (1992). A problem-solving model for episodic skeletan-plan refinement. *Knowledge Acquisition*, (4):197–216.

van Harmelen, F., (1991). *Meta-level Inference Systems.* Pitman, London.

Veloso, M., August 1992. *Learning by Analogical Reasoning in General Problem Solving.* PhD thesis, School of Computer Science, Carnegie Mellon University.

Verdaguer, A., July 1989. *PNEUMON-IA: Desenvolupament i Validació d'un Sistema Expert d'Ajuda al Diagnòstic Mèdic.* PhD thesis, Universitat Autònoma de Barcelona.

Verdaguer, A., Patak, A., Sancho, J., Sierra, C., and Sanz, F., (1992). Validation of the Medical Expert System PNEUMON-IA. *Computers and Biomedical Research*, 25(6).

Contents