CASE-BASED LEARNING OF STRATEGIC KNOWLEDGE

BEATRIZ LOPEZ ENRIC PLAZA
Institut d'Investigació en Intel·ligència Artificial
CEAB-CSIC, Camí de Santa Bàrbara, 17300 Blanes, Catalunya, Spain
bea@ceab.es, plaza@ceab.es

Research partially supported by CICYT 801/90 Massive Memory Project and Esprit II 2148 Valid Project.

Abstract

In this paper we describe BOLERO, a case-based reasoner that learns strategic knowledge (plans) to improve the problem-solving capabilities of an expert system. As a planner, BOLERO is a reactive planner that when gathering new observations can immediately generate a new plan to cope with the new situations. As a learner, BOLERO is capable of learning strategies from observation of the problem-solving performed by a teacher. From this experience, BOLERO plans strategies that solve new problems. BOLERO learns from success and failure during its problem-solving process. An evaluation to measure the efficiency of BOLERO is performed by comparing the system's results against both the correct solution of a case and the solutions provided by different domain experts. BOLERO has been proved useful in acquiring strategic knowledge and in refining existing strategies, in a real-life expert system for pneumonia diagnosis.

Keywords: Strategy learning, case-based learning, planing, control knowledge.

1. Introduction

When approaching the application of machine learning techniques to the knowledge acquisition process of expert systems it is important to know what are the main problems today for building expert systems. In a recent study developed at Carnegie Group on the building of expert systems the surprising result was that the major effort (50%) of knowledge engineers was dealing with the uncontrolable interaction among rules and assuring that the proper sequence of goals and rule chainings is achieved during problem solving (Carbonell 1990). In this paper we show how this issue can be solved through the automatic acquisition of strategies that control the behavior of a rule-base. Strategies are acquired by case-based learning techniques by observing the problem-solving behavior of an expert and by learning from the system's own case-based problem-solving.

In this paper we describe BOLERO, a system that learns strategies to improve solving capabilities in expert systems. Problem solving strategies avoid to explore the whole search space for a problem,

focus the process to the most likely solutions (as human experts are able to do), and reduce the number of observations or questions made to the user (low cost solution). Thus learning strategies means to learn efficient ways of solving problems.

Usually strategies are present in expert systems as control knowledge. Control knowledge is in charge of the goal/subgoal decomposition of a task, the detection of an end-condition, etc. For example, in pneumonia diagnosis, to validate the hypothesis *anemia* control knowledge splits the search in several subgoals as *hypoproliferates*, *hyperproliferates*, and *maturing defect*. Control knowledge can be explicitly separated from domain knowledge (e.g. using meta-level rules as in TEIRESIAS (Davis, 84), MILORD (Godo et al., 89), and MU (Cohen, 87)) or implicitly coded in the ordering of rules. The aim of BOLERO is to provide an expert system with the capability of learning a strategic knowledge base, in order to control the execution of the problem solving of that expert system. An overview of BOLERO is shown in figure 1. Learning in BOLERO has two modes: the training mode and the performance mode. In the training mode BOLERO learns strategies from observation of a teacher solving a set of problems. The goal of the training mode is that BOLERO acquires an initial set of strategies from a teacher. Learned strategies are organised in a dynamic memory as we will see in section 2.

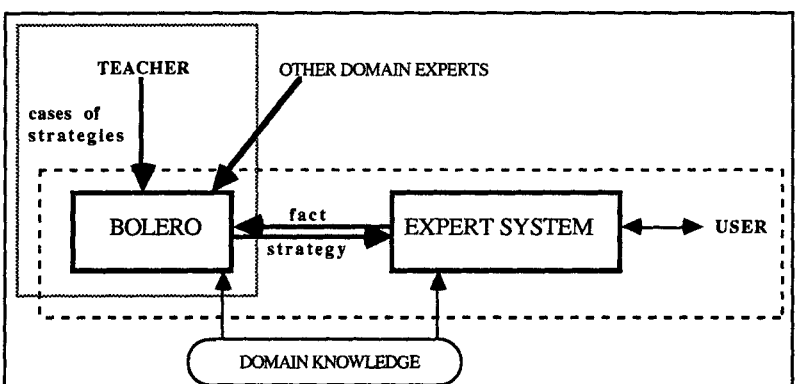

Fig.1. Overview of BOLERO. The dotted box contains the elements involved in the training mode. The dashed box contains the elements involved in the performance mode.

During the performance mode, BOLERO builds up strategies to solve a new problem. A new strategy is the result of the combination of one or more strategies of cases in memory. Strategies are constructed dynamically, according to the observable data provided for the current problem, and they are executed by an expert system[1]. Two new opportunities to learn are possible in the performance mode when solving a problem: (i) when BOLERO succeeds in solving the current problem and incorporates it in memory for future use, and (ii) when the strategy fails to achieve a good solution. The performance mode is described in section 3.

[1]The expert system only contains domain knowledge, and no strategic knowledge whatsoever is provided. The architecture of the expert system (based on MILORD [Godo et al., 89]) is linked to BOLERO via a plan interpreter that schedules the problem solving of the domain knowledge sources accordingly.

The performance mode needs some criteria to know how good is a strategy (solution) planned by BOLERO. This has lead us to develop an evaluation process where BOLERO results are compared against the teacher results as well as other domain experts. With this multi-expert evaluation we avoid to mimetize the teacher's behavior. BOLERO's evaluation is described in section 4. Finally in section 5 we sketch some implementation considerations and in section 6 a discussion of the system is performed.

1.1. BOLERO framework

The objects that play the main role in BOLERO are strategies, and cases. An strategy is represented by a sequence of goals considered along the problem solving and that lead to a problem solution. A case is a sequence of steps (problem states) performed to solve a problem. A step $S(t)$ consists of the observed facts $F(t)$ known up to time t, and the current goal $G(t)$ the problem solver (human or machine) is considering at time t given $F(t)$ (see table 1). The problem solver chooses, at each step, which information to ask or gather from the environment such that is the most useful for achieving the most recent goal $G(t-1)$. Figure 2 summarises a case using two conventions: first, only the steps where the current goal is changed are shown, and second, only the new data gathered during the interval in which a given goal is current are shown.

$$S(t) = \langle F(t), G(t) \rangle$$

$$F(t) = \{f \mid f \in \text{Facts and } f \text{ is known at time } t\}$$

$$F(t+1) \supset F(t) \qquad |F(t+1)| = |F(t)| + 1$$

$$G(t) \in \text{Goals}$$

Table 1. Definition of a step

As an example, case *C05v1* of figure 2 is a case of pneumonia diagnosis, where in step 2 the goal *pneumococcus* was considered, and in step 3 *injured lobes* was asked, allowing the teacher to conclude the goal: *(pneumococcus very-possible)*. Immediately, the teacher focuses on a new goal, namely *enterobacteriaceae*, as suggested by the data available up to *t=3* (see step 3 in figure 2). A special step 0 is required in order to start the diagnostic process. Step 0 contains a default goal (the *acquire-general-data* goal, used to start gathering data from the user). We will distinguish between *training-cases* and *problem-cases*. Training-cases are problems already solved (i. e.past experiences). Problem-cases are cases being solved.

Two assumptions should be made before using BOLERO. The first one is that we assume that a domain knowledge base (DKB) is already existing in the expert system for which BOLERO learns strategic knowledge. No new learning is performed on domain knowledge and the DKB organisation is known and used by BOLERO in both training and performance modes. The second one is that we have a kind of oracle, as for example a teacher, a domain expert, a book, etc., that can be used by BOLERO as a model of performance.

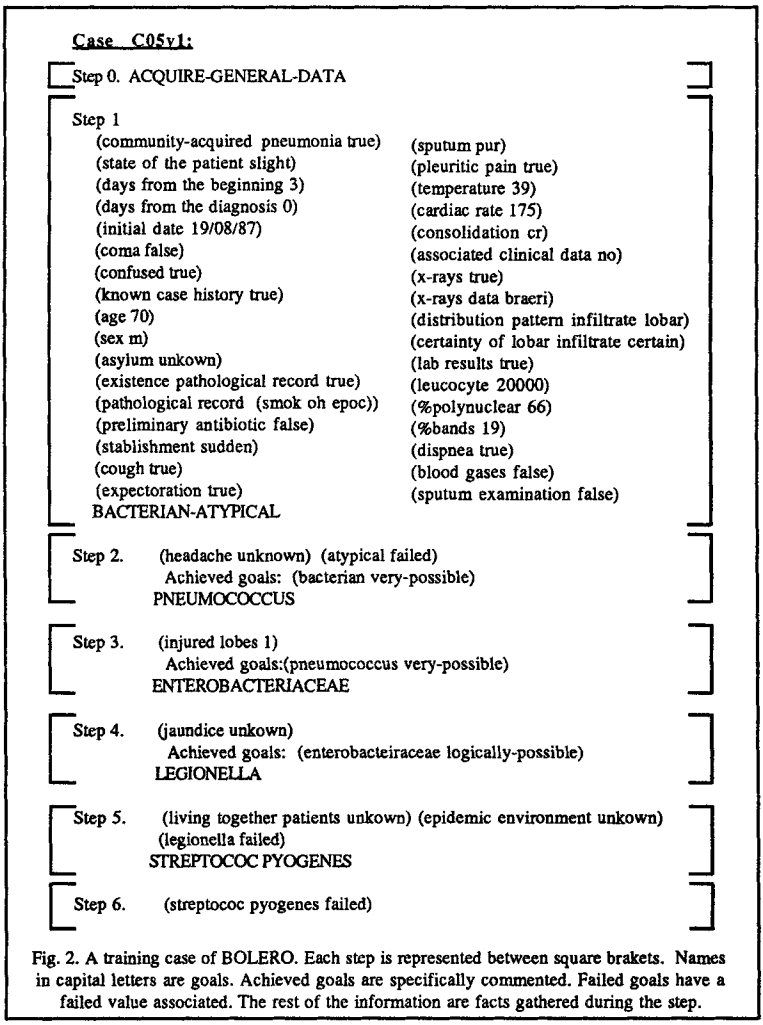

Fig. 2. A training case of BOLERO. Each step is represented between square brakets. Names in capital letters are goals. Achieved goals are specifically commented. Failed goals have a failed value associated. The rest of the information are facts gathered during the step.

2. Learning by observation

The training mode aim is to provide a knowledge base of strategies to BOLERO. The training set consists of training-cases. Training-cases are organised in BOLERO's memory as a tree where each node keeps information about a step, i. e. Node(i) = <F(i), G(i)>. Training-cases are split along the tree in such a way that to recover them it is necessary to follow the paths from the root to the leaves of the tree (see figure 3). A strategy associated with a node is the sequence of goals found in the path from the root to that node. For example, the strategy associated with the node <N5> of figure 3 is the sequence of the goals from *acquired-general-data* goal to the goal *legionella* stored in node <N5>, i. e. (*acquire-general-data, bacterian-atipical, pneumococcus. enterobacteriaceae, legionella*). Facts of nodes are used as indexes to

remind nodes from facts.These indexes will play an important role when retrieving past experiences to solve new problems (see section 3.1).

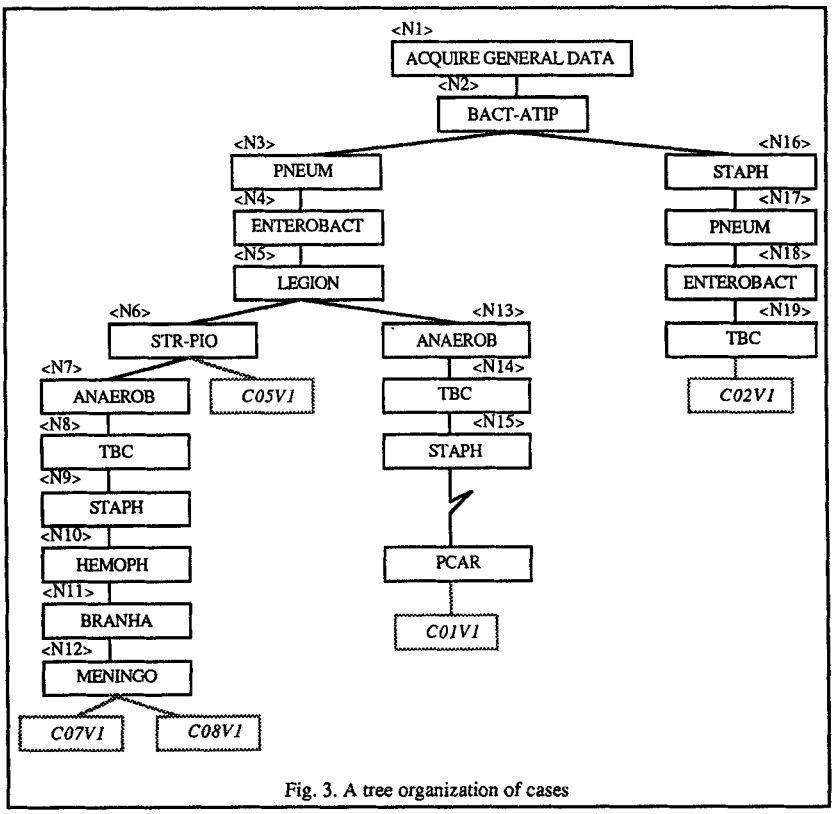

Fig. 3. A tree organization of cases

The learning by observation consists of two main processes: the incorporation process and the generalisation process. The incorporation process is in charge of the incorporation of a single training-case in memory, re-organizing the current memory. The generalisation process helps the incorporation process when different facts present in the training-case and other nodes need to be generalised. Before processing any training-case the tree is empty. During learning by observation, the incorporation of training-cases is performed incrementally training-case by training-case, for a set of training-cases provided by the teacher.

2.1 Incorporation process

BOLERO incorporates the problem solving it observes from a training-case into the memory tree. Before incorporating a training-case, BOLERO analyses the strategy in order to assure that goals only appears once along the strategy. Very often goals are tried but abandoned because some evidence focuses the problem solving to a different goal and afterwards these goals are retrieved and achieved. The incorporation process rationalises the sequences of goals, erasing duplicate goals and maintaining each goal in the last place where it occurs. Failed goals are kept in the strategy because their failure may be

relevant. The result of this rationalisation is the strategy incorporated to the system's memory.For example suppose that the original strategy of a training-case is the following: (g_a, g_b, g_a , g_c , g_d). g_a is tried, g_b is tried and achieved, g_a is tried again and achieved this time,g_c is tried and failed, and g_d is tried and achieved. Then BOLERO rationalises the strategy as (g_b, g_a , g_c , g_d) and incorporates it to the memory as a training case. [2]

```
Procedure Incorporate-Case(C)
Let C be the current training-case with steps {S1, ... Sk}
     where Si=<Fc(i), Gc(i)>
Let N={Root}
Let i=1
While ∃ n ∈ N such that Gc(i)=Gn(i) for n=<Fn(i), Gn(i)>
     Let Fn(i) = Generalize(Fc(i),Fn(i))
     Let i = i+ 1
     Let N = sons(n)
Let n' = Create-Node(parent(p), Si) where p∈N
∀j, k≥j≥i+1
     Let n'=Create-Node(n', Sj)

Procedure Create-Node(parent, Sj)
     Let M be a new node  with <Fc(i), Gc(i)>
     Let Parent(M)=parent
     Return M
```

Table 2. Incorporation of a training-case in memory

A training-case is added to the tree by successively comparing the goal of its *i*-step against any node at the *i*-level of the tree. If no node *n* (of the *i* -level) has a goal that matches the goal of the *i*-step, then the facts of the *i*-step and the facts of the node *n* are generalised, and the result is updated in the node *n*. Then the process continues with the step *i+1* of the training-case and the nodes in the level *i+1* of the memory tree the ancestor of which is the node *n*. This process is performed for all steps of the current training-case. If any node is found that matches the goal of step *i* of the training-case, then a new branch of the tree is started from the level *i-1*, and the rest of the steps of the training-case become new nodes from level *i* to level *k* where step *k* is the last step of the tree. Each new node is created from a step of the training-case. In table 2 the algorithm is shown.

2.2 Generalisation process

The construction of the memory tree involves a generalisation process such that facts stored on each node are the result of a generalisation of the facts of the nodes that are placed under that node. The generalisation process is based on the hierarchies defined in the domain knowledge that establish different kinds of relations between facts. For example the facts *cardiocirculatory clinical data*, , *hematopoietical clinical data, neurological clinical data, cutaneous scars* (and others) are of type *associated clinical data*

[2]The rationalization process reduces the case storage in memory since different strategies are mapped to the same memory representation. Nevertheless, this simplification does not affect the capability of generating new strategies which restart abanded goals due to BOLERO's reactive planning, as explained in section 4.1.

and can be generalised to *(associated clinical data true)*. Values of facts are generalised depending on its type. Integers are generalised by using intervals relevant to the application. The generalisation of two enumerated values is the union of the values. The generalisation of two uncertain values is the lowest value. Most of the generalisations are performed as in the ARC system (Plaza & López de Mantaras, 90).

3. BOLERO's problem solving

Once BOLERO has some knowledge about strategies in the memory tree, the system is applied to solve new problems. As a case-based reasoner, the basic mechanism of BOLERO is as follows:

0. Start with a default strategy (the strategy associated to the root of the tree) until new evidence is found.
1. Retrieve from memory the nodes more similar to the current situation (i. e. the facts known up to a time t of the current problem-case). If the retrieved nodes coincide with the last retrieved nodes then go to 3.
2. Construct a new strategy by adapting the strategies of the retrieved nodes.
3. Execute the constructed strategy onto the problem-case (until $t+\partial$ where new evidence is achieved), then go to 1.
4. Stop when the current strategy has been completed.

Notice that BOLERO acts as a reactive planner: anytime the current situation changes, the strategy executed by the system can be interrupted by the activation of new nodes in memory which suggest a better strategy, the new strategy is then constructed and executed immediately. When solving problems, BOLERO acts as the control knowledge component of the cycle of execution of an expert system. It proposes to the expert system a strategy to follow, and the expert system executes it until a new fact is obtained. BOLERO analyzes the new fact and decides whether to proceed with the same strategy or change to a new one.

BOLERO's problem solving strongly depends on how the retrieval of nodes are performed and how the strategies of the retrieved nodes are adapted to solve the current problem-case. The following subsections explain how retrieval and strategy adaptation is performed.

3.1 Retrieval

The retrieval of nodes is performed by matching the known facts of the current problem-case $F(t)$ against facts stored in nodes. This matching is performed incremented for the most recently obtained fact (*current fact*). The matching is the same as the matching of the generalisation process in the training mode. Facts are indexes to nodes, as has been stated in (section 2), and this allows BOLERO to retrieve only the nodes that contain the current-fact or the facts that match the current fact. The matching score for a node is computed as follows:

1. For each fact that exactly matches the current fact in a node it scores 1.

```
1.   Let A = {e1, e2, ..., en} such that
     ∀ ei, score(ei) > threshold (threshold is given) and
     ∀ ei, ej, if  i < j, then score(ei) > score(ej).
3.If e1 is a node, and
          ∀ ei ∈ A, i > 1, score(e1) >> score(ei)
     then return strategy(e1)
4.   Otherwise,
     Let B = {e1, e2, ..., eß} where ß is given.
     If ∃ a node nk, such that
          ∀ ei ∈ B, strategy(ei) ⊃ strategy(nk)
     then return strategy(nk).
5.   Otherwise return strategy(e1).
```

Table 3. Adaptation of a new strategy

2. A score λ is assigned to facts of a node that partially match the current fact (Plaza & López de Mántaras, 90)
3. Finally for all nodes activated a normalised total score is computed based on the scores of their facts.

In order to elaborate the current strategy only one node or few ones are chosen to construct the strategy. This second selection is based on the following criteria:(a) select a node if it has a score far higher than the rest, and (b) if neither one is found, then select only the first ß nodes (ß is given)

3.2 Strategy adaptation

Once BOLERO has one or ß nodes most similar to the current problem-case, it constructs the strategy adapting the strategies of the selected nodes. Adaptation is performed as follows: if there is only one node selected, the strategy of such node is applied to the current problem-case. Otherwise BOLERO looks for a common ancestor for the ß nodes in the memory tree. The strategy of the node identified as the lowest common ancestor is taken as a partial strategy to be followed in the current stage of problem-solving. However, the strategy of the common ancestor may be already executed, and therefore there is no guarantee that the common ancestor can propose a strategy useful for the current stage of problem-solving. If a useful common ancestor cannot be found, then the strategy of the highest score node is applied. Table 3 sketches the algorithm followed.

This process is performed for each new fact available for the current problem-case. So a different and better strategy can be decided anytime the system obtains new information. The problem solving process stops when the selected strategy finishes executing the current problem-case. Then the expert system gives a *domain solution* (e. g. the solution shown in figure 4). This domain solution has been achieved following a strategy that we will call from now on the *solution strategy*.

Once BOLERO has solved the current problem-case, it is incorporated in the memory organisation of cases by using the incorporation process described in section 2.1, as it was a training-case. If the problem-case has been solved according to a strategy of a node already in memory, this incorporation will

cause only new generalisations of facts attached to nodes. However BOLERO's knowledge of strategies is refined when the solution strategy was not already in memory.

Solution set for case B13v1	
Bacterian pneumonia	*Quite possible*
Pneumococcus pneumonia	*Possible*
Clamydia	*Slightly possible*
Hemophylus	*Very low chance*
Tuberculosis	*Very low chance*

Fig. 4. A domain solution of the expert system

4. Learning by BOLERO´s own experience

BOLERO learns from success and failure through strategic memory refinement. To know the success of BOLERO performance we have developed an evaluation process that will be described in section 4. Refinement from success occurs when the solution of the problem-case is successful and the solution strategy is different of any other in memory. Refinement from failure occurs when the domain solution of the problem is rejected in the evaluation process.

4.1 Learning from success

As we have seen, BOLERO solves a new problem by dynamically adapting strategies of nodes according to the data known at each problem solving state. As a result of this reactive planning, the strategy *Str(C)* constructed for solving a problem-case *C* is the sequence of the effective followed parts of the strategies BOLERO generated during problem solving. This emergent strategy *Str(C)*, although based on past strategies, can be a *novel strategy* (i. e. *Str(C)* is different from any past strategy in memory). BOLERO can therefore learn new strategies from its own successful problem solving when including the new strategies in memory.

For instance, suppose that in a given moment BOLERO´s strategic knowledge is represented as in figure 5(a). A new problem-case *Cn* is being solved. At the beginning a strong match was produced against node *<N5>* (corresponding to the training-case *C1*), but progressively the match against node *<N9>* (corresponding to the training-case *C2*) increases, and the final execution of the case *Cn* follows that of node *<N5>* (fig 5(b)). Finally the execution of case *Cn* is completed and the strategy followed by *Cn* is the sequence of parts of the strategies of both nodes, *<N5>* and *<N9>*. That is *(acquire-general-data, bacterian-atypical, pneumococcus, enterobacteriaceae)* from *<N5>* and *(staphilococcus, tuberculosis)* from *<N9>*. Note that when a change of strategy is performed, goals already achieved are not repeated (in the example: *pneumococcus* and *enterobacteriaceae*). So in the problem-case *Cn* BOLERO has followed a strategy not explicitly known before. Therefore when the problem-case *Cn* is incorporated in memory, BOLERO´s organisation of past experiences is refined as shown in figure 5(c).

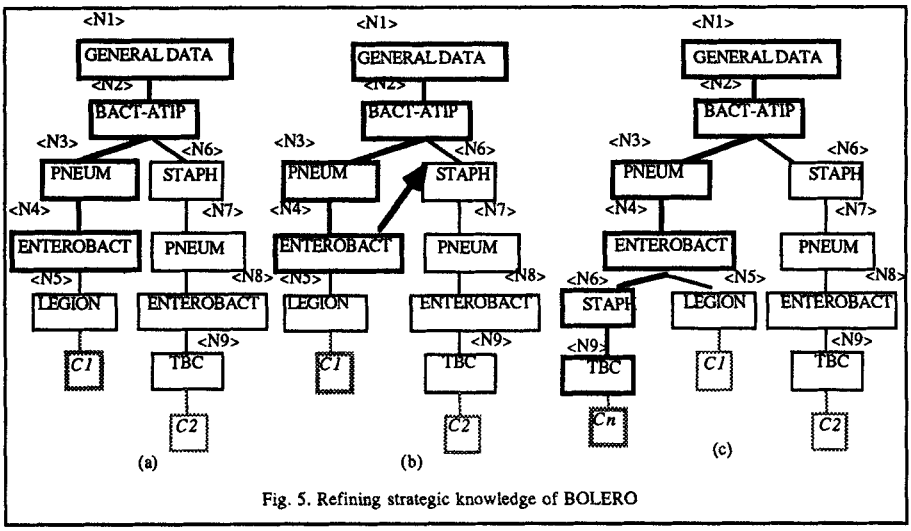

Fig. 5. Refining strategic knowledge of BOLERO

4.2 Learning from failure

The second opportunity to learn arises when the solution of the current problem-case is rejected. A strategy is evaluated as a bad solution and rejected, as we will see in section 5, when a set of goals that should have been pursued were not considered in the solution strategy. BOLERO has two options to correct the failure and avoid it in the future. The first option is using the training mode: BOLERO asks to the teacher for the correct strategy solution and the problem-case is incorporated in memory. The second option is goal exploration: BOLERO continues generating a new strategy for achieving goals that have not yet been tried but should have been tried and achieved. BOLERO's method to learn from failure is to select a strategy from the set of nodes activated in the performance mode but whose strategy has not yet been executed for the current problem-case. A new strategy is generated and BOLERO tries to prove new goals until the solution is accepted by the evaluation process. When the solution strategy is accepted it is incorporated in BOLERO's memory. Although the new strategies generated by BOLERO are based on past experience, the search performed (i. e. the goals that are pursued) is less focussed in goal exploration than usual. For this reason, when incorporating the problem-case into memory, the goals that have been tried but not successfully established after the failure are not incorporated into memory as part of the problem-solving strategy of the current problem-case. Note that BOLERO assumes domain knowledge in DKB correct, i. e., goals achieved by the solution strategy cannot be false positives. Although the system does not store the negative results of failing it learns from failure in the sense that repairs the wrong solution (using the teacher advice or thru goal exploration) and therefore prevents those failures to occur again.

5. Evaluation of BOLERO's strategies

As we stated in the introduction, BOLERO has some criteria to evaluate the domain solution of the current problem-case. One solution could be to ask to the teacher for his opinion. However this method will lead to mimetize the teacher's behavior. Our approach consists in providing BOLERO with a gold standard with which it can compare its results with the help of a teacher. The gold standard can be either the known correct domain solution or different solutions given by several domain experts. The first kind of gold standards is not always available. In pneumonia diagnosis, for example, the correct solution is rarely known. In fact, diagnoses are known only when there is conclusive laboratory data that isolates the pathogenic agent[3]. The second kind of gold standard involves consulting different experts opinions to evaluate the results of the problem-cases solved by BOLERO.

The evaluation process consists in comparing the solution provided by BOLERO against the gold standard. First of all, the success or failure of BOLERO's solution is assessed. If successful, we also evaluate the focusing degree of the solution strategy. Success in problem solving is assessed by two conditions: (1) if the correct domain solution is available, BOLERO's solution is correct when it includes all the domain solutions established by the gold standard[4], and (2) also it is mandatory that the set of achieved goals of the solution strategy provided by BOLERO includes the achieved goals of the solution strategy suggested by the teacher.

Once BOLERO gets past this first stage of the evaluation, the system's solution is accepted and it is possible to measure the focusing degree of the solution strategy. We can have different degrees of focussing depending on the solution strategy of BOLERO:

- Loss of focusing: when BOLERO has considered some goals not considered by any expert and no result is obtained, then we can say that BOLERO has lost focusing
- Averaged solution: when the domain solutions proposed by BOLERO are not identical to those established by the teacher, but the goals not present in the solution have been considered by other domain experts, then we can say that the solution strategy of BOLERO is an averaged solution.

6. Implementation

The first application chosen to develop BOLERO is a pneumonia diagnosis expert system, Pneumon-IA, built using the MILORD shell[5].. The pneumonia application has been implemented by a

---

[3]But most of the clinical cases of the application we have use to develop BOLERO do not contain this data, because most patients can be cured by expert physicians without lab tests, which are expensive and slow. The set of cases with correct solution is highly biased towards dangerous cases due to a complicated evolution of patients.

[4] We assume that the expert system to which BOLERO plans strategies deals with uncertain information. So instead of a unique solution, the result of the system is a set of possible solutions with different degrees certainty (see for a example fig. 4).

[5]Briefly described MILORD is an expert system shell whose main features are its multilevel architecture and its uncertainty management method [Godo et al., 89].The modularity of having separate

knowledge engineer along all the different knowledge levels of MILORD. We have taken Pneumon-IA as a whole and used it as a teacher for BOLERO. Moreover, when we disable the strategic knowledge of Pneumon-IA, the expert system is useful to test the strategies generated by BOLERO. Pneumon-IA was validated in 1989 by comparing its results against other five domain experts and Pneumon-IA scored as the second best expert among them (Verdaguer, 89). Of the 86 cases the results of 10 cases where exactly known due to the patients lab tests. The evaluation criteria described in section 5 has been applied using the 86 pneumonia cases and the diagnoses of the five expert physicians on all those cases.

BOLERO has been developed using PCL in a SUN machine. The mechanism of indexing in BOLERO is implemented using a spreading-activation mechanism that simulates the parallelism of marker-passing techniques (see Kolodner, 87). Our future research agenda includes implementing the parallelism of marker-passing techniques in an hypercube IPSC/2 multicomputer for memory search and retrieval.

Table 4 shows some evaluation results on the behavior of BOLERO after testing 10 problem-cases. We can say that BOLERO performance is as good as Pneumon-IA but tens to be less focused. That is, BOLERO learns to consider all the goals Pneumon-IA would have pursued (see first row in Table 4). Moreover sometimes BOLERO considers some goals not pursued in Pneumon-IA. Some are irrelevant but but others were indeed considered by other experts, and whose processing have produced positive results (row 2 in Table 4). The exact goal ordering used in Pneumon-IA is not respected by BOLERO; but they are generally very close in their execution order, so the change of order does not have meaningful consequences.

Training Cases	16	36	46	76
Successes	40%	40%	40%	90%
Missing goals	2.33	2.16	2	2
Averaged Solution	30%	20%	20%	30%
Averaged goals	2.66	3	2	1.3
Loss of focus	5.62	5.10	4.50	4.31

Table 4. Evaluation of BOLERO for four suites of training cases and 10 problem-cases. Success measures the cases in which BOLERO achieves the correct domain solution. Missing goals are the mean number of missed goals in a failed case. Averaged solution measures the number cases where bolero achived goals relevant to other experts but not considered by the teacher. Averaged goals are the mean number of those goals. Loss of focus measures the number of goals in a case considered by BOLERO but could not be achieved.The averaged length of the teacher solution provided for the same 10 problem-cases is 17,3 goals. Pneumon-IA has a total of 35 goals.

---

levels for different kinds of knowledge (strategic & plan levels for strategic knowledge, metarules & domain levels for domain knowledge) allows the connection of BOLERO 's strategic reasoning and learning with the domain level of MILORD in a clean way.

7. Conclusion and discussion

Most of the methods developed in machine learning deal with concept learning. In this paper, however, we introduce a system that learns strategies to solve problems in expert systems. A previous work on acquiring strategic knowledge is ASK (Gruber, 89), an acquisition tool to elicit strategic knowledge from experts. Although ASK learns strategic rules, the kernel of ASK is based on an interactive dialogue between the system and the domain expert. Besides, ASK has been proved efficient when refining an existing strategic knowledge base but it is hard to build strategic knowledge from scratch. The method to acquire strategic knowledge we propose is useful both for learning strategies from scratch and for refining strategies already learned as we have seen along the paper. Moreover, BOLERO learns from success and failures while ASK only learns from failure. Learning control knowledge has also been done using machine learning techniques like explanation-based learning (EBL) in (Minton, Carbonell, Etizioni, Knoblock & Kuokka, 87). Although there is much research in EBL with imperfect theories, EBL seems to impose strict conditions on the type of knowledge needed. Our case-based approach is more germane to AI research on planing and does not require any of those strong requirements.

We can view strategies as plans, and from this perspective we can relate BOLERO with the CHEF system (Hammond, 89) CHEF is a case-based planning system whose aim is to reuse plans. CHEF learns by correctly indexing its planning experiences in memory as BOLERO does. Both systems generalise the features used to index cases in order to make plans applicable in similar but not identical situations. However, the mechanism of indexing in CHEF is based on a pre-selected group of features, while in BOLERO all features are used. Another important difference between CHEF and BOLERO is that the former constructs one-shot plans. Instead, BOLERO is a reactive planner: as gathering new observations the system can generate a new plan to cope with the new situation. This difference stems from the need of CHEF to work only with complete information (i. e. CHEF starts planning only when all information is available to the system). BOLERO is instead designed to plan in uncertain environments: information is usually incomplete, and the information is gathered (as specified by domain knowledge) only if it is relevant for the goals the system has decided to be worth pursuing.

In the future we plan to apply BOLERO to the learning of strategies for a rheumatology diagnosis expert system. We are also thinking to use the prototypicality of exemplars defined in PROTOS and ARC (Plaza & López de Mantaras., 90) in the definition of the matching function of the performance mode. Another improvement we are planning is to identify sub-strategies, that is, a very similar (but not exact) sequence of goals that appears quite often as a part of different strategies.

ACKNOWLEDGEMENTS

We are indebted with Ramon López de Mántaras and Walter van de Welde who red previous versions of this paper, providing a lot of useful suggestions and comments. Research partially supported by CICYT 801/90 Massive Memory Project and Esprit II 2148 Valid Project.

References

Bareiss E.R., Porter B.W.& Wier C.C. Protos: An exemplar-based learning apprentice, *Proc. 4th International Workshop on Machine Learning*, University of California, Irvine, 1987.
Carbonell J., Personal Communication, 1990.
Cohen P.R. The control of reasoning under uncertainty: A discussion of some programs, COINS Technical Report 87-81, University of Massachusetts, Amherst, 1987.
Davis R. Interactive transfer of expertise, *Rule-Based Expert Systems.*, Edited by B.G. Buchanan & E.H. Shortliffe, Addison-Wesley, 1984.
Godo L., López de Màntaras R., Sierra C.& Verdaguer A. MILORD: The architecture and the management of linguistically expressed uncertainty, *International Journal of Intelligent Systems*, Vol. 4, Num. 4, Winter 1989.
Gruber T.R. *The acquisition of strategic knowledge*, Academic Press, Inc., 1989.
Hammond K.J. *Case-Based Planning*. Academic Press Inc., 1989.
Kodratoff Y.& Ganascia J.G. Improving the generalization step in learning, *Machine Learning*, Vol. II, Morgan Kaufmann Pub., 1986.

Kolodner J.L. Retrieving events from a case memory: A parallel implementation, *DARPA Workshop on Case-Based Reasoning*, 1987.

Minton S., Carbonell J.G., Etzioni O., Knoblock C.A.& Kuokka D.R. Acquiring effective search control rules: Explanation-based learning in the PRODIGY system, *Proc. 4th International Workshop on Machine Learning*, University of California, Irvine, 1987.

Plaza E.& López de Màntaras R. ARC: Learning typicality knowledge from fuzzy examples, In Ras, Zemankova, Emrich (Eds.), *Methodologies for Intelligent systems 5*, p. 420-427. North-Holland, 1990.

Verdaguer, A. *PNEUMON-IA: Desenvolupament i validació d'un sistema expert d'ajuda al diagnòstic mèdic*, PhD. Thesis, Universitat Autònoma de Barcelona, July 1989.