Dynamic Case Base Maintenance for a Case-Based Reasoning system

Maria Salamó and Elisabet Golobardes

Enginyeria i Arquitectura La Salle, Universitat Ramon Llull,
Quatre Camins 2, 08022 Barcelona, Catalonia, Spain
{mariasal,elisabet}@salleurl.edu

Abstract. The success of a case-based reasoning system depends critically on the relevance of the case base. Much current CBR research focuses on how to compact and refine the contents of a case base at two stages, acquisition or learning, along the problem solving process. Although the two stages are closely related, there is few research on using strategies at both stages at the same time. This paper presents a model that allows to update itself dynamically taking information from the learning process. Different policies has been applied to test the model. Several experiments show its effectiveness in different domains from the UCI repository.

1 Introduction

Learning is a process in which an organized representation of experience is constructed [Scott, 1983]. However, this experience cause two problems in Case-Based Reasoning (CBR) systems, as reported in recent years. The first one is the *swamping problem* which relates to the expense of searching large case-bases for appropriate cases with which to solve the current problem. The second one is that the experience can be *harmful* and may degrade the system performance (understanding performance as problem solving efficiency).

Research on the area highlights to deal with negative knowledge using different strategies. Negative Knowledge is correct knowledge that can be a source of unsuccessful performance [Markovitch, S. and Scott, P.D., 1988]. Minton has demonstrated by *selective discarding knowledge* in a system [Minton, 1985] that the performance can be improved. Usually, the strategy of avoiding negative knowledge in the initial case base is not enough to achieve maximum performance for a CBR system. It is usually also necessary to integrate into the system a repeated maintenance during the problem solving process. There are several methods that fulfill these requirements, like competence-preserving deletion [Smyth and Keane, 1995], failure-driven deletion [Portinale et al., 1999], as well as for generating compact case memories [Smyth and Mckenna, 2001]. More close to our proposal are the one that examines the benefits of using fine-grained performance metrics to guide case addition or deletion [Leake and Wilson, 2000].

Previously to this paper, we have presented different approaches to case base maintenance [Salamó and Golobardes, 2003] in acquisition stage that allow us to reduce the case base in a controlled way and, at the same time, maintain the efficiency in the CBR system. Although our objectives had been achieved, our previous conclusions and the research on the area move us to go deeply into an extended treatment of the case base.

This paper introduces a dynamic case base maintenance (DCBM) model that updates the knowledge (case base in CBR) based on the learning problem solving process. The knowledge update is based on Reinforcement Learning. This approach can be considered as a ”wrapper” model to case base maintenance. However, the authors propose it as a dynamic model because it depends completely on the problem solving process of the CBR system.

The paper is organized as follows. In section 2 we introduce the dynamic case base maintenance model and then different policies to apply it. Section 3 details the fundamentals of our experiments. Next section shows and analyzes the effectiveness of the model with the experimental results. Finally, we present the conclusions and further work.

2 Dynamic Case Base Maintenance

The foundation of our Dynamic Case Base Maintenance (DCBM) proposal is Reinforcement Learning. So, first we summarize its basis. Next, we describe how to use the Reinforcement Learning in our system, how the coverage of a CBR system can be modelled, and how different policies can exploit this model to perform a dynamic experience update able to control and optimize the case base while solving new cases.

2.1 Reinforcement Learning

Reinforcement Learning (RL) [Sutton and Barto, 1998] combines the fields of dynamic programming and supervised learning to yield powerful machine-learning systems. Reinforcement Learning appeals to many researchers because of its generality.

Reinforcement Learning [Harmon, 1996] is an approach to learning by trial and error in order to achieve a goal. A RL algorithm does not use a set of instances which show the desired input/output response, as do *supervised learning* techniques. Instead, a reward given by the environment is required. This reward evaluates the current state of the environment. The *Reinforcement Learning Problem* (RLP) consists of maximizing the sum of future rewards. The goal to be accomplished by RL is encoded in the received reward. To solve the problem, a RL algorithm acts over the environment in order to yield maximum rewards. Any algorithm able to solve the RLP is considered a RL algorithm.

Reinforcement Learning theory is usually based on *Finite Markov Decision Processes* (FMDP). The use of FMDP allows a mathematical formulation of the RLP, therefore, the suitability of RL algorithms can be mathematically proved.

Several elements appear in all RLPs. In each iteration the RL algorithm observes the *state* s_t of the environment and receives the *reward* r_t. The reward is a scalar value generated by the *reinforcement function* which evaluates the current state and/or the last executed action according to RLP. Following the rules of the RL algorithm, it generates an *action* a_t. The *environment* reacts to the action changing state s_t and generating a new state s_{t+1} . The *value function* contains the expected sum of future rewards. This function is used and modified by the RL algorithm to learn the policy function. A *policy function* indicates the action to be taken at each moment.

Initially, the approximation of the optimal value function is poor. Therefore, it is necessary to approximate the value function at every iteration. There are several methods that can be applied.

In order to find the optimal value functions, the Bellman equation is applied: $V^*(X_t) = r(X_t) + \gamma V^*(X_{t+1})$, where $V^*(X_t)$ is the optimal value function; X_t is the state vector at time t; X_{t+1} is the state factor vector at time $t+1$; $r(X_t)$ is the reinforcement function and γ is the discount factor in the range $[0, 1]$.

2.2 Dynamic case base maintenance model

There are several methodologies to solve the RLP formulated as a FMDP: dynamic programming, temporal difference algorithms and monte-carlo methods. We will use a Monte-Carlo method because is the only one that use experience of the environment to learn the value functions.

The question that arises now is how this idea can be applied to our model. Lets consider the model by analogy of the elements described in section 2.1. For our purpose a *state* s_t is a *case* of the environment that receives a *reward* r_t. The reward is a value generated by the *reinforcement function* which evaluates if the current state classifies or not classifies correctly. In our model the *reinforcement function* is the revise phase of the CBR cycle. Following the rules of the RL algorithm, which includes the case base maintenance policy, it generates an *action* a_t. The action for us is to delete or to maintain a case from the case base. The *environment* is the CBR cycle. The *environment* reacts to the action changing to state s_{t+1}, if the action is to delete the case. Thus, reducing the case base. The environment also generates a new reward after the problem solving process which has used the possibly reduced case base. The *value function* contains the expected sum of future rewards. This function is used and modified by the RL algorithm to learn the optimal case base. We test two different policy functions. Figure 1 shows the description of all the process. In our case, the RL algorithm receives a set of states and a reward for each one, and returns to the environment a set of actions.

Definition 1 (Coverage)
Let $T = \{t_1, t_2, ..., t_n\}$ be a set of training cases, $\forall\ t_i \in T$: $Coverage_k(t_i)$ will be the value of the metric used by the case base maintenance method at iteration k.

The *coverage* is the goodness value of a case when it is used to solve a target problem. It can be defined in several ways depending on the case base maintenance techniques used. For instance, it can be defined [Smyth and Keane, 1995] as the set of target problems that it can be used to solve. Here, we modify slightly the definition in order to adapt it to our model. The *coverage* is defined as the initial sum of future rewards using a Rough Sets measure. That is, $Coverage_k(t_i)$ is the value function at iteration k for state t_i.

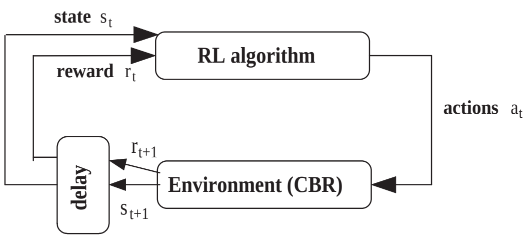

Fig. 1. Relation between RL algorithm and the environment.

As detailed previously, the most important part of the RL algorithm is to update the value function. We use a Monte-Carlo (MC) which interacts with the environment following a particular policy function. In our model it is the optimizer of the case base. When the episode finishes, the MC algorithm updates the value of all visited states based on the received rewards. The visited states for a CBR cycle will be the kNN cases retrieved to solve the new problem. Equation 1 shows the general update rule to estimate the state-value function. Our MC algorithm is detailed in definition 2.

Definition 2 (CoverageUpdate)
Let $T = \{t_1, t_2, ..., t_n\}$ be a set of KNN cases, $\forall\ t_i \in T$:

$$Coverage_{k+1}(t_i) \leftarrow Coverage_k(t_i) + \alpha \cdot |R_t - Coverage_k(t_i)| \qquad (1)$$

It can be observed that the current prediction of the state-value $Coverage_k(t_i)$ is modified according to the received sum of rewards R_t. The R_t value is 1.0 if the state t_i solve the target problem, otherwise it is 0. There is also a learning rate α which averages the values obtained in different episodes. The learning rate is usually set up to value 0.1 or 0.2 in RL systems. If the states are updated quite often it is set up to value 0.1, otherwise to 0.2. The selection of KNN neighbors in a CBR cycle may not often be repeated, so we have set up this learning rate to 0.2 in order to accelerate the differences of *Coverage* in few iterations.

Once described our value function update, we describe entirely the dynamic case base maintenance (DCBM) model in algorithm 1, which shows that the retrieval phase selects K Nearest-Neighbors, although it uses the best neighbor to solve the new problem. We consider the selection of KNN in order to accelerate the maintenance process of the case base. Another important point is the relation of the retain stage with the RL algorithm (step 9 and 10) in algorithm 1. The retain phase receives the set of actions to improve the case base.

The most notable aspect of the dynamic case base maintenance process is that the CBR system improves the case base using its problem solving process. Moreover, the case base improves or degrades the coverage of a case depending

```
Algorithm 1 Dynamic Case Base Maintenance (DCBM) model

 DCBM (CaseMemory $T$)
  1. Initialize $Coverage(t_i)$ using a CBM metric in acquisition stage, for all $t_i \in T$
  2. $T_{k+1} \leftarrow$ Reduce the initial case base $T_k$ using $Coverage$
  3. Repeat until problem solving process of the CBR cycle is not finished
  4. $T_k \leftarrow T_{k+1}$
  5.   Retrieval phase $\leftarrow$ selects from $T_k$ the $K$NN used to solve the new problem
  6.   Reuse phase $\leftarrow$ selects the best $1NN$ to solve the new problem
  7.   Revise phase $\leftarrow$ computes the rewards $R_t$ of the $K$NN
  8.   Retain phase $\leftarrow$ computes :
  9.     CoverageUpdate $\leftarrow$ for each $t_i \in K$NN
 10.     Apply case base maintenance policy function to decide the set of Actions $A$
 11.     $T_{k+1} \leftarrow$ Update case base $T_k$ based on the Actions $A$
```

on their resolution accuracy. Thus, the case base can be categorized at different levels of coverage. The lower the coverage of a case, the most appropriate to disappear from the case base.

2.3 Dynamic Case base Maintenance policy functions

The core of the RL process is the case base maintenance policy function. We describe two different policies to test the reliability of the proposed Dynamic Case Base Maintenance (DCBM) model.

RLOLevel This policy is called Reinforcement Learning Oblivion policy by Level of Coverage (RLOLevel). This policy uses a similar philosophy that our acquisition [Salamó and Golobardes, 2003] case base maintenance method called ACCM. If we start from the premise that ACCM works well to reduce the case base while maintaining the prediction accuracy, it leads us to believe that the same process will be useful for dynamic maintenance. Thus, the complete process is detailed in algorithm 2.

```
Algorithm 2 RLOLevel

 1. SelectCasesRLOLevel (CaseMemory $T$)
 2. confidenceLevel = 1.0 and freeLevel = ConstantTuned (set at 0.01)
 3. select all instances $t_i \in T$ as $SelectCase(t_i)$ if $t_i$ satisfies:
    $coverage(t) \geq$ confidenceLevel
 4. while not $\exists$ at least a $t_i$ in $SelectCase$ for each class $c$ that $class(t_i) = c$
 5.   confidenceLevel = confidenceLevel - freeLevel
 6.   select all instances $t_i \in T$ as $SelectCase(t_i)$ if $t_i$ satisfies:
      $coverage(t_i) \geq$ confidenceLevel
 7. end while
 8. Action $A$ is to delete from CaseMemory $T$ the set of cases NOT selected as $SelectCase$
 9. return Action $A$
```

The algorithm 2 tries to remove as much cases as possible. Therefore, the selection process is repeated until it accomplishes that every distribution class contains at least one case selected. Thus removing from case base those cases not selected. It is clear that this process will be very aggressive with the case base because it maintains the minimum description of the case base. It leads us to believe that this policy function may not work properly in a dynamic environment.

RLOCE This policy is called Reinforcement Learning Oblivion by Coverage and Error (RLOCE). The coverage is the relevance of a case. This policy shows the simplest way to decide the actions.

Algorithm 3 RLOCE

```
1. SelectCasesRLOCE (CaseMemory $T$)
2. for each instance $t \in T$
3.   if $coverage(t) < initialCoverage(t)$ then $SelectCase(t)$ end if
4. Action $A$ is to delete those cases selected
5. return Action $A$
```

The policy is based on coverage lost. A case will be deleted if it classifies incorrectly new problems more often than correctly. Thus, the cases that produce misconception are deleted.

3 Description of the experimental analysis

This section is structured as follows: first of all, it is important to understand the fundamentals of our metric to initialize the coverage of a case. Then, we describe the testbed used and its characteristics. Finally, we analyze with different experiments the dynamic case base maintenance model.

3.1 Fundamentals

The rough sets theory defined by Pawlak, which is well detailed in [Pawlak, 1982], is one of the techniques for the identification and recognition of common patterns in data, especially in the case of uncertain and incomplete data. The mathematical foundations of this method are based on the set approximation of the classification space.

Each case is classified using the elementary set of features which can not be split up any further, although other elementary sets of features may exist. In the rough set model the classification knowledge (the model of the knowledge) is represented by an equivalence relation IND defined on a certain universe of cases U and relations (attributes) R. The pair of the universe cases U and the associated equivalence relation IND forms an approximation space. The approximation space gives an approximate description of any subset X of U. Two approximations are generated by the available data about the elements of the set X, called the lower and upper approximations. The *lower approximation* $\underline{R}X$ is the set of all elements of U which can *certainly* be classified as elements of X in knowledge R. The *upper approximation* $\overline{R}X$ is the set of elements of U which can *possibly* be classified as elements of X, employing knowledge R. In order to discover patterns of knowledge we should look for the minimal set of attributes that discerns cases and classes from each other, such a combination is called a *reduct*.

Measure of relevance based on Rough Sets The reduced space, composed by the set of *reducts* (P) is used as a metric to extract the relevance of each case.

Definition 3 (Coverage based on Rough Sets)
This metric uses the *quality of classification* coefficient, computed as:

$$\forall\ t_i\ \in\ T\ it\ computes: Coverage(t_i) = \frac{card\ (\ \underline{P}(t_i))\ \cup\ card\ (-\overline{P}(t_i))}{card\ (\ all\ instances)} \tag{2}$$

Where $Coverage(t_i)$ will be the coverage of case t_i; T is the training set; $card$ is the cardinality of a set; P is a set that contains the reducts; and finally $\underline{P}(t_i)$ and $\overline{P}(t_i)$ is the presence of t_i in the lower and upper approximation respectively.

The $Coverage$ coefficient expresses the percentage of cases which can be correctly classified employing the knowledge t. This coefficient has a range of real values in the interval [0.0, 1.0]. Where 0.0 and 1.0 mean that the case is internal and outlier respectively.

We will use the $Coverage$ as $initialCoverage$ in our DCBM model. We also use the $Coverage$ in our reduction technique (ACCM) in acquisition stage. Our experiments analyze the behaviour of DCBM model in front of ACCM. Our RLOLevel policy function is based on this algorithm. We apply ACCM in the training case base to select a range of cases that have to be deleted from the case base [Salamó and Golobardes, 2003]. ACCM maintains all the cases that are outliers, so cases with a $Coverage$ = 1.0 value, and those cases that are completely internal, so cases with a $Coverage$ near 0.0. Thus, reducing from the case base those cases that are not outlier and have a coverage near 1.0.

Using *coverage* values, we have two kind of cases relevant in the case base: the ones with coverage value of 1.0 (outliers) and the internal cases, having low coverage value. This coverage distribution is not much suitable for the RL policy functions which rely on high coverage values. Thus, we modify, previously to update phase and independently if we have applied ACCM or not, the *coverage* value with this formula: $Coverage(t) = 1 - Coverage(t)$, with the exception of outlier cases that have a $Coverage(t) = 1.0$. Therefore, we obtain *coverage* values that show relevance according to RL policy functions.

3.2 Testbed

The evaluation performance of the approaches presented in this paper is done using different datasets which are detailed in table 1. Datasets can be grouped in: *public* [Merz and Murphy, 1998] and *private* [Golobardes et al., 2002] that comes from our own repository. These datasets were chosen in order to provide a wide variety of sizes, combinations of feature types, and difficulty because some of them contain a great percentage of inconsistencies.

The percentage of correct classifications and the percentage of case base maintained has been **averaged** over **stratified ten-fold cross-validation** runs. To study the performance we use **paired *t-test*** on these runs.

Table 1. Details of the datasets used in the experimental analysis

	Dataset	Ref.	Samples	Num. feat.	Sym. feat.	Classes	%Inconsistent
1	*Balance scale*	*BL*	625	4	3	2	2.0
2	*Breast cancer Wisconsin*	*BC*	699	9	-	2	0.30
3	*Credit-A*	*CA*	690	5	9	2	9.71
4	*Heart-H*	*HH*	294	6	7	5	20.4
5	*Heart-Statlog*	*HS*	270	13	-	2	0.0
6	*Hepatitis*	*HP*	155	6	13	2	0.0
7	*Horse-Colic*	*HC*	368	7	15	2	5.67
8	*Ionosphere*	*IO*	351	34	-	2	0.0
9	*Iris*	*IR*	150	4	-	3	0.0
10	*Labor*	*LB*	57	8	8	2	0.0
11	*Mammogram (private)*	*MA*	216	23	-	2	5.00
12	*Soybean*	*SY*	683	-	35	19	10.08
13	*TAO-Grid (private)*	*TG*	1888	2	-	2	0.0
14	*Vehicle*	*VE*	846	18	-	4	0.0
15	*Vote*	*VT*	435	-	16	2	4.13

The study described in this paper was carried out in the context of our CBR system: BASTIAN (*case*-**BA***sed* **S***ys***T***em for class***I***fic***A***tio***N**). All techniques were run using the same set of parameters for all datasets: The case base is a list of cases. Each case contains the set of attributes, the class, the *Coverage* and the *initialCoverage*. Furthermore, the retrieval phase extracts the K-Nearest Neighbor to be updated in the RL process, not for the reuse phase which uses a **1-Nearest Neighbor**. We do not learn new cases during problem solving stage.

4 Analysing the DCBM policy functions

First of all, we test our DCBM policy functions using all the training set in front of 1NN algorithm and our reduction algorithm (ACCM) in acquisition stage (see columns 2 to 9 in table 2). We introduce in this experiment ACCM algorithm in order to compare the case base reduction (size) with our DCBM policies.

We observe that the best prediction accuracy is often obtained using oblivion by level of coverage (OL) and oblivion by coverage and error (OCE). Looking at ACCM algorithm, it has greater reduction than 1NN. In spite of the fact the reduction of DCBM policies is not as great as ACCM, because its selection to delete is founded on the KNN selected, they produce a good balance between reduction and improvement of prediction accuracy. That is, they are less aggressive reducing the case base than ACCM.

There is a clear conclusion: if we prefer to reduce the case base while maintaining the prediction accuracy of the system, it is better to use DCBM model than ACCM applied only during acquisition. Once analysed DCBM model alone, we test the combination between acquisition (ACCM) and learning (DCBM) stages at the same time.

Table 2 shows (from column 10 to 15) the results of such combination. In this case, the ACCM final case base will be the initial one for the DCBM policies. Before examining this question in detail, let us notice that there are two results to highlight: the percentage of cases maintained by our DCBM policies and the final case base size when finishing both processes. The percentage of cases maintained during oblivion (**obliv**) is computed using this formula $\frac{\#finalcases}{\#finalcasesACCM} \times 100$, which shows the behavior of the DCBM policies. The percentage of final case base size (**size**) shows the percentage of case base maintained from the original training set, it is computed using this formula $\frac{\#finalcases}{\#traincases} \times 100$.

Table 2. Results for all methods using an update parameter KNN = 5 Av1 shows the mean value for all datasets. We use paired t-test at the level of 5% significance, where a ● and a ○ stand for a significant improvement or degradation of DCBM policies and ACCM to 1NN

Ref	**cbr**		**cbm**		**cbr**		**cbr**		**cbm**			**cbm**		
	1NN	*size*	**ACCM**	**size**	**OL**	*size*	**OCE**	*size*	**OL**	*obliv*	*size*	**OCE**	*obliv*	*size*
BL	76.15	100.0	77.27	●97.44	78.73	●88.69	78.73	●88.69	78.11	85.89	83.69	79.04	●90.25	87.94
BC	95.86	100.0	95.43	77.36	95.99	67.93	95.99	97.61	**96.26**	38.79	30.01	95.98	96.67	74.78
HC	73.36	100.0	70.91	○86.14	**81.24**	●88.79	**81.24**	●88.79	**81.24**	●81.49	70.19	80.16	●88.53	76.26
CA	81.76	100.0	82.19	84.30	82.63	89.40	82.63	89.40	82.47	86.80	73.17	82.47	87.35	73.63
MA	63.93	100.0	64.53	89.19	62.11	51.44	63.39	77.77	55.88	○ 7.55	6.73	**64.91**	78.95	70.42
TG	96.13	100.0	96.13	95.87	**96.66**	97.44	**96.66**	97.44	63.92	○ 0.23	0.22	96.60	97.72	93.69
HH	72.82	100.0	72.12	85.63	75.56	●87.86	75.56	●87.86	75.19	14.38	12.32	**76.23**	●88.12	75.47
HS	74.07	100.0	75.55	79.67	74.81	86.74	74.81	86.74	75.18	29.28	23.33	**77.03**	85.27	67.94
HP	77.99	100.0	77.33	87.67	78.58	87.67	78.58	87.67	74.75	47.83	41.93	77.87	86.09	75.48
IO	86.92	100.0	87.20	83.79	87.74	91.45	87.74	91.45	**88.01**	54.25	45.45	87.74	91.16	76.38
IR	95.33	100.0	**96.66**	89.03	95.33	97.03	95.33	97.03	91.33	○ 8.56	7.63	96.00	97.60	86.96
LB	83.38	100.0	83.04	77.38	**87.04**	88.50	**87.04**	87.91	81.14	52.14	40.35	86.47	86.65	67.05
SY	82.15	100.0	83.83	●78.38	87.15	●92.09	**87.28**	●91.65	86.22	●87.96	68.94	86.22	●88.58	69.43
VE	69.43	100.0	68.13	72.36	69.53	80.33	69.53	80.33	68.36	67.40	48.77	68.38	72.23	52.27
VT	86.65	100.0	90.78	●79.23	92.60	●95.47	92.60	●95.47	91.96	●40.39	32.00	**92.86**	●95.03	75.30
Av	81.06	100.0	81.40	84.22	**83.04**	86.05	**83.14**	89.72	79.33	46.82	38.98	**83.19**	88.68	74.86

We concentrate on different observations in table 2 that allow us to express points in favour of the DCBM model.

- Reduction of the case base during the acquisition stage is not enough. As the results show in ACCM column and we have also noticed previously, it is necessary to delete "harmful" knowledge during the problem solving process.
- The DCBM using the problem solving process helps the system to obtain a more accurate and reduced case base.
- The reduction obtained using DCBM augment the prediction accuracy of standard 1NN algorithm, with the exception of the combination between ACCM and OL. The combination does not work because it is too much aggressive with the case base, as expected previously when defined.
- On the other hand, OL works properly if it is not combined with ACCM, even though it has a great reduction policy to select the cases for being removed from the case base. In conclusion, OL can be only applied alone.
- The combination of ACCM with OCE does not improve often the performance of OCE applied alone. However, the combination has a higher reduction than OCE alone and also improves on average previous prediction accuracy.

5 Conclusions

This paper proposes a model for case base maintenance that uses the dynamics of the problem solving process to search for the optimal case base while maintaining the prediction accuracy. The experimental study demonstrates that the DCBM model using different policies manage to get the initial objectives: it optimizes the case base while it improves on average the prediction accuracy of the system. Our further work will be focused on testing the model in recommender systems in order to analyze a dynamic environment with our dynamic model. We also think of testing different case reduction methods on acquisition stage.

Acknowledgements This work is supported by the *Ministerio de Ciencia y Tecnologia*, Grant No. TIC2002-04160-C02-02. We wish to thank *Enginyeria i Arquitectura La Salle - Ramon Llull University* for their support to our Research Group.

References

[Golobardes et al., 2002] Golobardes, E., Llorà, X., Salamó, M., and Martí, J. (2002). Computer Aided Diagnosis with Case-Based Reasoning and Genetic Algorithms. *Knowledge-Based Systems*, (15):45–52.

[Harmon, 1996] Harmon, M. (1996). Reinforcement learning: A tutorial.

[Leake and Wilson, 2000] Leake, D. and Wilson, D. (2000). Remembering Why to Remember: Performance-Guided Case-Base Maintenance. In *Proceedings of the Fifth European Workshop on Case-Based Reasoning*, pages 161–172.

[Markovitch, S. and Scott, P.D., 1988] Markovitch, S. and Scott, P.D. (1988). The Role of Forgetting in Learning. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 459–465.

[Merz and Murphy, 1998] Merz, C. J. and Murphy, P. M. (1998). UCI Repository for Machine Learning Data-Bases [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. *Irvine, CA: University of California, Department of Information and Computer Science.*

[Minton, 1985] Minton, S. (1985). Selectively generalizing plans for problem solving. In *Ninth International Joint Conference on Artificial Intelligence*, pages 596–599. Morgan Kaufmann.

[Pawlak, 1982] Pawlak, Z. (1982). Rough Sets. In *International Journal of Information and Computer Science*, volume 11.

[Portinale et al., 1999] Portinale, L., Torasso, P., and Tavano, P. (1999). Speed-up, quality and competence in multi-modal reasoning. In *Proceedings of the Third International Conference on Case-Based Reasoning*, pages 303–317.

[Salamó and Golobardes, 2003] Salamó, M. and Golobardes, E. (2003). Hybrid Deletion Policies for Case Base Maintenance. In *Proc. of the sixteenth International FLAIRS Conference*, pages 150–154. AAAI Press.

[Scott, 1983] Scott, P. (1983). Learning: The construction of a posteriori knowledge structures. In *Proceedings of the Third National Conference on Artificial Intelligence.*

[Smyth and Keane, 1995] Smyth, B. and Keane, M. (1995). Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In *Proceedings of the Thirteen International Joint Conference on Artificial Intelligence*, pages 377–382.

[Smyth and Mckenna, 2001] Smyth, B. and Mckenna, E. (2001). Competence Models and the maintenance problem. *Computational Intelligence*, 17(2):235–249.

[Sutton and Barto, 1998] Sutton, R. and Barto, A. (1998). *Reinforcement Learning. An introduction.* The MIT Press.