Dynamic update of experience for a Case-Based Reasoning system

Maria Salamó and Elisabet Golobardes
Enginyeria i Arquitectura La Salle, Universitat Ramon Llull,
Quatre Camins, 2 - 08022 Barcelona, Spain

Abstract. Experience is one of the major factor of success of Case-Based Reasoning systems. This paper presents a learning algorithm that introduces reminding to update dynamically the experience of a system using a Reinforcement Learning model. Current research focuses on maintaining the experience growth by applying reduction techniques, but usually they do not consider adding new experience. For this reason, we propose a learning algorithm combined with two oblivion algorithms. All the algorithms are integrated into our model. Several experiments show the effectiveness of all the approaches in different domains from the UCI repository.

1 Introduction

Experience in a Case-Based Reasoning (CBR) system is known as case base. The success of CBR system depends critically on the relevance of the case base. In recent years, two problems have been addressed in research related to the case base. 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). Different techniques, known as case base maintenance techniques, have been related to solve both problems. However, few approaches are concerned about the relationship between adding experience and reduction strategies.

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]. One common solution is to use forgetting strategies. Minton has also demonstrated by *selective discarding knowledge* in a system [Minton, 1985] that the performance can be improved. It is usually also necessary to integrate into the system a repeated case base maintenance during the problem solving process. There are several methods that fulfill these requirements, like competence-preserving deletion [Smyth and Keane, 1995] and failure-driven deletion [Portinale et al., 1999], as well as for generating compact case memories [Smyth and Mckenna, 2001] through competence-based [Zhu and Yang, 1999] case addition. More close to our proposal are the one that examines the benefits of using fine-grained performance metrics to directly guide case addition or deletion [Leake and Wilson, 2000] and the one that integrates the case base maintenance process [Reinartz and Iglezakis, 2001] with the overall case-based reasoning process.

Previously to this paper, we have presented different approaches to case base maintenance [Salamó and Golobardes, 2003] that allow us to reduce the case base in a controlled way and,

at the same time, maintain the efficiency in the CBR system. However, research on the area move us to go deeply into an extended treatment of the experience inside the CBR system.

This paper presents a learning algorithm combined with two oblivion algorithms based on a Dynamic Case Base Maintenance (DCBM) model that update experience dynamically. The experience update is based on Reinforcement Learning. This approach can be considered as a "wrapper" to case base maintenance. However, the authors propose it as a dynamic update model because it depends completely on the problem solving process of the CBR system.

This paper is organized as follows. Section 2 introduces the dynamic case base maintenance model and then the learning algorithm combined with two oblivion strategies. Section 3 details the fundamentals of our experiments and analyzes the effectiveness of the algorithms. Finally, we present the conclusions and further work.

2 Dynamic Case Base Maintenance model

The foundation of our learning algorithm and oblivion algorithms are a Dynamic Case Base Maintenance (DCBM) model based on Reinforcement Learning. So, first we summarize the basis of Reinforcement Learning. Next, we describe how to use it in our system, how the coverage of a CBR system can be modelled, and how the different algorithms exploit this model to perform a dynamic experience update able to control and optimize the case base growth while introducing 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 to *state* s_t and generating *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 in each 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 introduced into 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 three 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.

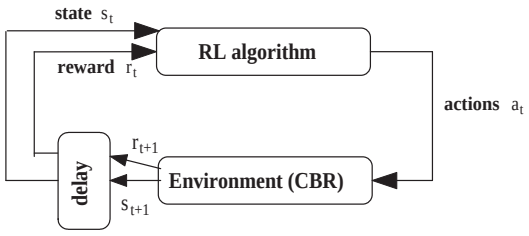

Figure 1: Relation between RL algorithm and the environment.

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.

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 2.2.

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

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

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 0.0. 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.

Algorithm 1 **Dynamic Case Base Maintenance (DCBM)**

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 KNN 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 KNN
8. Retain phase $\leftarrow$ computes :
9. CoverageUpdate $\leftarrow$ for each $t_i \in K$NN
10. **Learning algorithm** to decide if new case is added
11. Apply case base maintenance **policy function** to decide the set of Actions A
12. $T_{k+1} \leftarrow$ Update case base T_k based on the Actions A

Once described our case base coverage, we describe entirely the dynamic case base maintenance model in algorithm 1, which shows that the retrieval phase selects K Nearest Neighbors, although it uses 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 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 Learning based on Reinforcement Learning (LearnRL)

Learning is a process in which an organized representation of experience is constructed [Scott, 1983]. The case base constructed is dynamic and it is necessary having into account previous generation changes. Algorithm 2 summarizes the process of the proposed learning algorithm that lets the case base to add cases if knowledge is incomplete and proportionates to them the generational experience.

Algorithm 2 **Learning based on Reinforcement Learning (LearnRL)**

LearnRL (CaseMemory T, CaseMemory KNN, Case $newCase$, Case $retrievedCase$)

1. **if** $retrievedCase$ solve correctly $newCase$ **then** store = false
2. **else if** there exists a case in KNN that solve correctly $newCase$ **then** Selects the most similar as $auxiliarCase$ and store = true
3. **else if** there exists a case in T that solve correctly $newCase$ **then** Selects the most similar as $auxiliarCase$ and store = true
4. **end if**
5. **end if**
6. **end if**
7. **if** store **then** // update the information of the $newCase$ using the experience of the $auxiliarCase$
8. $newCase.Coverage$ = $auxiliarCase.Coverage$
9. $newCase.InitialPrecision$ = $auxiliarCase.Coverage$
10. **Add** $newCase$ in T
11. **end if**
12. **return** CaseMemory T

When a new experience takes place, it is not simply added to the case base of prior experiences. Most experiences are like others that have come before. Algorithm 2 is similar in its foundations to the *Condensed Nearest Neighbor* rule [Hart, 1968]. If a case is misclassified using the case base, it is added to it. This rule is applied to avoid further misclassification. Second part of algorithm 2 is to proportionate generational experience to the case that is being added. The algorithm finds experiences that are most closely related to the input case ($newCase$) we are processing. In this algorithm, relatedness is defined by nearest neighbors. Generational experience can be seen as the phenomenon of *reminding*.

Reminding is a highly significant phenomenon that has much to say to us about the nature of memory [Schank, 1982]. It tells us about memory organization. It also tells us about learning and generalization. In our case, reminding reveals something significant about the nature of memory structures and the understanding process, shown in algorithm 2 as $Coverage$. The equation 3 defines the nature of the $Coverage$. In this case, it is based on *Rough Sets* theory.

2.4 Dynamic Case base Maintenance policy functions

Reminding is at the root of how we understand. It is also at the root of how we learn. However, we also forget experiences that do not use when learning.

The RL process described in algorithm 1 illustrates how to update the reminding of the system. The process to forget is enclosed into the case base maintenance policy function. We describe two different policies that have been combined with our *Learning algorithm* to test the reliability of the proposed algorithm of reminding.

2.4.1 RLOC

This policy is called Reinforcement Learning Oblivion by difference of Coverage (RLOC). The coverage is the relevance of a case. RLOC uses the coverage of each case in a different way, trying to reduce the computational cost of the previous policy function. Algorithm 3 shows the simple policy function applied in RLOC.

Algorithm 3 **Reinforcement Learning Oblivion by difference of Coverage (RLOC)**

1. RLOC (CaseMemory T)
2. **for** each instance $t_i \in T$
3. $SelectCase(t_i)$ if t_i satisfies:
$$\frac{initialCoverage(t_i) - coverage(t_i)}{initialCoverage(t_i)} \geq 0.20 \quad (2)$$
4. **end for**
5. **Action** A **is to delete** those cases **selected**
6. **return** Action A

RLOC is quite fast to compute. By computing the balanced difference between its initial coverage and the updated coverage, we obtain a metric of the behavior of the case. If it is positive, the case produce a misconception when solving a new problem. The percentage of misconception allowed for a case is set up to value 0.2 (this is a lost of 20% coverage). This percentage lets a case classify incorrectly new cases only two times.

2.4.2 RLOCE

This policy is called Reinforcement Learning Oblivion by coverage and error (RLOCE). This policy shows the simplest way to decide the actions.

Algorithm 4 **RL Oblivion by Coverage and Error (RLOCE)**

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

The policy is based also on coverage lost. However, here, a case will be deleted the first time it classifies incorrectly one case. The case has a second opportunity if it has classified correctly previously. Thus, the cases that produce misconception are deleted, with the exception of those cases that let the system classify correctly on some occasions.

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 proposed learning algorithm and its combination with 2 oblivion algorithms.

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

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

$$For\ each\ instance\ t_i\ \in\ T\ it\ computes:$$

$$Coverage(t_i) = \frac{card\ (\ \underline{P}(t_i))\ \cup\ card\ (\ -\overline{P}(t_i))}{card\ (\ all\ instances)} \tag{3}$$

Where $Coverage(t_i)$ will be the coverage of the instance 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 $Coverage(t_i)$ 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. The higher the quality, the nearer to the outlier region.

We will use the *coverage* as *initialCoverage* in our DCBM model. Using *coverage* values, we have two kind of cases relevant in the case base: the ones with coverage value of 1.0 (outlier) and the internal cases, having low coverage values. This coverage distribution is not much suitable for the RL policy functions relying on high coverage value. Thus, we convert previously to update phase the $coverage$ value with this formula: $Coverage(t)' = 1 - Coverage(t)$, with the exception of those cases that have a $Coverage(t) = 1.0$.

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,] 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.

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 represented as 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**.

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.

3.3 Analyzing LearnRL algorithm and the combination with OC and OCE

To analyze our approaches, we test different algorithms: (1) 1-Nearest Neighbor (1NN) with no learning taking place; (2) 1NN-L using as learning the CNN rule; (3) LearnRL (L-RL) algorithm; (4) LearnRL combined with oblivion policy OC and, finally LearnRL combined with OCE. Table 2 shows for each algorithm tested the prediction accuracy, percentage of final case base (size) when finishing the process computed as $\frac{train+stored-oblivion}{train+test}$ and the percentatge of test cases stored (ret) when retaining, it is computed as $\frac{stored}{test}$.

The results obtained by 1NN-L are the same as 1NN which shows that the introduction of new experience in the CBR system does not help to improve prediction accuracy (PA). LearnRL obtains a results very similar to 1NN-L although it improves when retaining is smaller. The most important part of LearnRL is the acquisition of generational experience into the cases added to the system. These experience is used by the oblivion policies to update dynamically the case base. The combination between LearnRL with OC and OCE improves most often than 1NN algorithm. Moreover, the final case base size is considerably smaller than 1NN. The reduction of case base is due to 2 factors: (1) the algorithm stores few number of cases because it classifies correctly new experiences; (2) the oblivion procedure uses generational experience to decide which cases to forget from the case base. We observe that the best PA is often obtained by LearnRL combined with OCE, because OCE after a misconception of a case analyzes if it is necessary to delete it from the case base. Nevertheless, OC is more conservative, thus needing a longer problem solving process. However, all the results improve, so they lead us to validate that the generational experience learned by the model lets the system to improve dynamically its case base when solving new problems.

Table 2: Results for all methods using an update parameter KNN = 5. The mean value (Av) for all datasets is at the bottom part of the table. We use paired t-test at the level of 5% significance, where a • and a ○ stand for a significant improvement or degradation of 1NN using learning (1NN-L), LearnRL (L-RL), LearnRL combined with OC (OC) and LearnRL combined with OCE (OCE) to 1NN

Ref	**1NN**	**size**	**1NN-L**	**size**	**ret**	**L-RL**	**size**	**ret**	**OC**	**size**	**ret**	**OCE**	**size**	**ret**
BL	76.15	90.00	76.15	92.38	23.83	76.98	92.30	23.04	**78.41**•	86.80	21.60	**78.41**•	81.85	21.60
BC	95.86	90.00	95.86	90.41	4.14	95.86	90.44	4.43	95.55	89.97	4.43	**95.99**	88.19	4.00
HC	73.36	90.00	73.36	92.66	26.63	74.70	92.52	25.27	79.61•	89.83	20.38	**80.79**•	81.63	19.02
CA	81.76	90.00	81.76	91.87	18.69	81.76	91.87	18.69	81.76	90.14	18.26	**82.19**	82.24	17.82
MA	**63.93**	90.00	**63.93**	93.70	37.03	**63.93**	93.70	37.03	63.32	84.21	36.57	63.30	73.61	36.57
TG	96.13	90.00	96.13	90.38	3.86	96.45	90.35	3.54	96.29	89.41	3.70	**96.60**	88.03	3.39
HH	72.82	90.00	72.82	92.65	26.53	72.82	92.65	26.53	74.55•	87.92	25.51	**75.90**•	81.46	24.15
HS	74.07	90.00	74.07	92.51	25.18	74.07	92.51	25.18	**75.18**	87.18	24.81	74.07	80.55	25.92
HP	77.99	90.00	77.99	92.19	21.91	77.99	92.19	21.91	**78.66**	90.45	21.29	78.58	81.22	21.29
IO	86.92	90.00	86.92	91.31	13.10	86.92	91.31	13.10	**87.76**	88.97	12.53	87.48	83.56	12.53
IR	**95.33**	90.00	**95.33**	90.46	4.66	**95.33**	90.46	4.66	**95.33**	88.33	4.66	**95.33**	87.73	4.66
LB	83.38	90.00	83.38	91.75	17.54	83.38	91.75	17.54	85.38	88.24	15.78	**87.04**	80.17	14.02
SY	82.15	90.00	82.15	91.78	17.86	83.94	91.61	16.10	86.16•	90.05	13.90	**86.85**•	83.80	13.17
VE	69.43	90.00	69.43	93.06	30.61	69.43	93.06	30.61	69.54	88.55	30.37	**69.65**	75.35	30.26
VT	86.65	90.00	86.65	91.33	13.33	87.82	91.21	12.18	90.55•	89.49	9.42	**92.60**•	86.69	7.35
Av.	81.06	90.00	81.06	91.89	18.99	81.44	91.86	18.65	**82.53**	88.63	17.54	**82.98**	82.40	17.05

4 Conclusions

This paper proposes a learning algorithm that is able to introduce reminding while adding new experience in the CBR system. Reminding is used later to learn to forget ”harmlful” experience. The experimental analysis shows that the combination between LearnRL and oblivion policy functions manage to improve case base while augmenting on average the prediction accuracy. To sum up, reminding allows the system to add new information and to optimize their case base. Thus, maintaining and sometimes improving a significant degree of 5% the prediction accuracy. The lesson learned from the experiments is that *less is more*. It is necessary to add experience in a controlled way but it is also necessary to forget it. This

paper is the initial idea of introducing reminding in a CBR system and we think that there are too much to do. Our further work will be focused on using generational experience as relatedness between cases. We also think about testing different $coverage$ metrics to test the effectiveness of the model.

Acknowledgements This work is supported by the *Ministerio de Ciencia y Tecnologia*, Grant No. TIC2002-04160-C02-02. We wish to thank *Our University* for their support to our Research Group in Intelligent Systems.

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.

[Hart, 1968] Hart, P. (1968). The condensed nearest neighbour rule. *IEEE Transactions on Information Theory*, 14:515–516.

[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,] Merz, C. J. and Murphy, P. M. UCI Repository for Machine Learning Data-Bases [http://www.ics.uci.edu/∼mlearn/MLRepository.html].

[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.

[Reinartz and Iglezakis, 2001] Reinartz, T. and Iglezakis, I. (2001). Review and Restore for Case-Base Maintenance. *Computational Intelligence*, 17(2):214–234.

[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.

[Schank, 1982] Schank, R. (1982). Dynamic Memory: A Theory of Learning in Computers and People. *Cambridge, England: Cambridge University 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.

[Zhu and Yang, 1999] Zhu, J. and Yang, Q. (1999). Remembering to add: Competence-preserving case-addition policies for case base maintenance. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 234–239.