Justification-based Case Retention

Santiago Ontañón and Enric Plaza

IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia (Spain).
{santi,enric}@iiia.csic.es, http://www.iiia.csic.es

Abstract. A CBR system needs a good *case retention* strategy to decide which cases to incorporate into the case base in order to maximize the performance of the system. In this work we present a collaborative case retention strategy, designed for multiagent CBR systems, called the *Collaborative Case Bargaining* strategy. The CCB strategy is a bargaining mechanism in which each CBR agent tries to maximize the utility of the cases it retains. We will present a case utility measure called the *Justification-based Case Utility* (JCU) based upon the ability of the individual CBR agents to provide *justifications* of their own results. An empirical evaluation of the CCB strategy shows the benefits for CBR agents to use this strategy: individual and collective accuracy are increased while the size of the case bases is decreased.

1 Introduction

Obtaining a good case base is a main problem in Case Based Reasoning. The performance of any CBR system depends mainly in the contents of the case base. Therefore, maintaining compact and competent case base has become a main topic of CBR research. Empirical results have shown that storing every available case in the case base does not automatically improve the accuracy of a CBR system [7]. Therefore any CBR system needs a good *case retention* strategy to decide which cases to incorporate into the case base in order to maximize the performance of the system.

Our work focuses on *multiagent CBR systems* ($\mathcal{M}$AC) [6] where the agents are able to solve problems individually using CBR methods and where only local case bases are accessible to each individual agent. Problems to be solved by an agent can be sent by an external user or by another agent. The main issue is to find good collaboration strategies among CBR agents that can help improving classification accuracy both individually and collectively. In a previous work [4] we presented several strategies for collaborative case retention among groups of CBR agents that try to take advantage of being in a multiagent scenario. In this work we will present a new collaborative retention strategy called *Collaborative Case Bargaining* (CCB) and a case utility measure called *Justification-based Case Utility* (JCU).

The main difference between the new CCB strategy and the retention strategies in [4] is that we now present, a new measure for assessing the utility of

retaining a case. A case has a high utility value for a CBR agent if it can prevent the agent in making errors in the future, and a case has a low utility value if it will not contribute in reducing the number of errors that the agent will make in the future. Moreover, we also present a new way in which the CBR agents negotiate among them: the CCB strategy is a bargaining mechanism in which each agent tries to maximize the utility of the individually retained cases.

The *Justification-based Case Utility* (JCU) is based upon the ability of the individual CBR agents to provide *justifications* of their own results, i.e. that CBR agents are able to explain why they have classified a problem in a specific solution class. If a CBR agent is able to provide a justification for an incorrectly solved problem, this justification can be examined and try to prevent that the same error is made in the future. The Justification-based Case Utility does exactly this, and uses justifications to detect which cases can prevent a CBR agent to repeat an error in the future and assigns them higher utility values.

The structure of the paper is as follows. Section 2 gives the basic notions of multiagent CBR systems. Then, Section 3 introduces the concept of justifications in CBR systems. Section 4 explains in detail the Collaborative Case Bargaining retention strategy, including an illustrative example and discussion. Finally, Section 5 presents an empirical evaluation of the CCB strategy compared against some other case retention strategies. The paper closes with the conclusions section.

2 Multiagent CBR Systems

Formally, a $\mathcal{M}\mathsf{AC}$ system $\mathcal{M} = \{(A_i, C_i)\}_{i=1\ldots n}$ is composed on n agents, where each agent A_i has a case base C_i. In this framework we restrict ourselves to analytical tasks, i.e. tasks (like classification) where the solution is achieved by selecting from an enumerated set of solutions $K = \{S_1 \ldots S_K\}$. A case base $C_i = \{(P_j, S_k)\}_{j=1\ldots N}$ is a collection of problem/solution pairs. Each agent A_i is autonomous and has learning capabilities, i.e. each agent is able to collect autonomously new cases that can be incorporated to its local case base.

Moreover, since we focus on analytical tasks, there is no obvious decomposition of the problem in subtasks. When an agent A_i asks another agent A_j help to solve a problem the interaction protocol is as follows. First, A_i sends a problem description P to A_j. Second, after A_j has tried to solve P using its case base C_j, it sends back a message with a solution endorsement record.

Definition: A *solution endorsement record* (SER) is a record $\langle\{(S_k, E_k^j)\}, P, A_j\rangle$, where the collection of *endorsing pairs* (S_k, E_k^j) mean that the agent A_j has found E_k^j cases in case base C_j endorsing solution S_k—i.e. there are a number E_k^j of cases that are relevant (similar) for endorsing S_k as a solution for P. Each agent A_j is free to send one or more endorsing pairs in a SER record.

In our framework, collaboration among agents is done by using *collaboration strategies*. A collaboration strategy defines the way a group of agents can cooperate to jointly solve some task. In our framework, a collaboration strategy consist in an interaction protocol and a set of individual policies that the agents follow.

The interaction protocol determines the set of possible actions an agent can take in each moment. Each agent uses his individual policies to autonomously choose which of the possible actions to take at each moment is the best according to its individual goals and preferences.

The next section presents the *Committee* collaboration strategy, that the agents use in order to solve problems.

2.1 Committee Collaboration Strategy

In this collaboration strategy the agent members of a $\mathcal{M}\mathsf{AC}$ system $\mathcal{M}$ are viewed as a committee. An agent A_i that has to solve a problem P sends it to all the other agents in $\mathcal{M}$. Each agent A_j that has received P sends a solution endorsement record $\langle\{(S_k, E_k^j)\},\ P, A_j\rangle$ to A_i. The initiating agent A_i uses a voting scheme above upon all SERs, i.e. its own SER and the SERs of all the other agents in the multiagent system. The problem's solution is the class with maximum number of votes.

Since all the agents in a $\mathcal{M}\mathsf{AC}$ system are autonomous CBR agents, they will not have the same problem solving experience (in general, the cases in their case bases will not be the same). This makes it likely that the errors that each agent make in the solution of problems will not be very correlated, i.e. each agent will not err in the same problems. It is known in machine learning that the combination of the predictions made by several classifiers with uncorrelated errors improves over the individual accuracies of those classifiers [3] ("ensemble effect"). Thus, using the committee collaboration strategy an agent can increase its problem solving accuracy.

The principle behind the voting scheme is that the agents vote for solution classes depending on the number of cases they found endorsing those classes. However, we want to prevent an agent having an unbounded number of votes. Thus, we will define a normalization function so that each agent has one vote that can be for a unique solution class or fractionally assigned to a number of classes depending on the number of endorsing cases.

Formally, let $\mathcal{A}^t$ the set of agents that have submitted their SERs to the agent A_i for problem P. We will consider that $A_i \in \mathcal{A}^t$ and the result of A_i trying to solve P is also reified as a SER. The vote of an agent $A_j \in \mathcal{A}^t$ for class S_k is $Vote(S_k, A_j) = \frac{E_k^j}{c+\sum_{r=1\ldots K} E_r^j}$ where c is a constant that on our experiments is set to 1. It is easy to see that an agent can cast a fractional vote that is always less than 1. Aggregating the votes from different agents for a class S_k we have ballot for S_k as $Ballot^t(S_k, \mathcal{A}^t) = \sum_{A_j \in \mathcal{A}^t} Vote(S_k, A_j)$ and therefore the winning solution class is the class with more votes in total.

3 Justifications in Multiagent Systems

Many expert systems and CBR applications have an explanation component [9]. The explanation component is in charge of justifying why the system has pro-

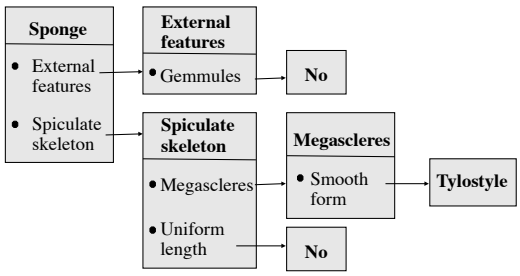

Fig. 1. Simbolic justification returned by LID.

vided a specific answer to the user. The line of reasoning of the system can then be examined by a human expert, thus increasing the reliability of the system.

All the existing work on explanation generation focuses on generating explanations to be provided to the user. However, in our approach we will use explanations (or justifications) as a tool for improving coordination among agents. Allowing the agents to give a justification of their individual results is crucial in multiagent systems since in an environment where one's conclusions may depend on knowledge provided by third parties, justifications of these conclusions become of prime importance [8]. In our work, we focus on individual agents that can provide justifications of their answers, and that can communicate those justifications to other agents. A CBR agent that receives a justification can then autonomously examine this justification in order to obtain information about the agent that created the justification. Moreover, we take benefit from the ability of some machine learning methods to provide more information than just the solution class, i.e. the ability to provide justifications.

Definition: A *justification* J built by a CBR method to solve a problem P that has been classified into a solution class S_k is a record that contains the relevant information that the problem P and the retrieved cases $C_1, ..., C_n$ (all belonging to class S_k have in common.

In our work, we use LID [2], a CBR method capable of building symbolic justifications. LID uses the feature term formalism to represent cases. *Feature Terms* (or ψ-terms) are a generalization of the first order terms. The main difference is that in first order terms (e.g. $person(barbara, john, dianne)$) the parameters of the terms are identified by position, while in a feature term the parameters (called *features*) are identified by name (e.g. $person[name \doteq barbara, father \doteq john, mother \doteq dianne]$). Another difference is that feature terms have a *sort*, for instance, the previous example belongs to the sort *person*. These sorts can have subsorts (e.g. *man* and *woman* are subsorts of *person*). Feature terms have an informational order relation ($\sqsubseteq$) among them called subsumption, where $\psi \sqsubseteq \psi'$ means all the information contained in ψ is also contained in ψ' (we say that ψ subsumes ψ'). When a feature term has no features (or all of its features are equal to $\perp$) it is called a *leaf*.

Figure 1 shows a symbolic justification returned by LID, represented as a feature term. Each box in the figure represents a node. On the top of a box the sort of the node is shown, and on the lower part, all the features with a known value are shown. The arrows mean that the feature on the left part of the arrow takes the node on the right as value. When LID returns this justification J for having classified a problem P in a specific solution class S_k, the meaning is that all the retrieved cases $C_1, ..., C_n$ by LID relevant for solving the problem P belong to the solution class S_k. The content of the justification J should be considered as a symbolic description of similarity, i.e. a description of what is common among $C_1, ..., C_n$ and P. Moreover, the symbolic similarity J contains the most relevant attributes of the problem P.

When an agent solves a problem the result is reified as a *justification endorsing record* (JER):

Definition: A *justified endorsing record* (JER) $\mathbf{J} = \langle P, S, J, A \rangle$ is a tuple containing the problem P, the solution class S found by the agent A for the problem P, and the justification J for that problem. (To refer to the elements of a JER, we will use the dot notation, i.e. we will use $\mathbf{J}.J$ to refer to the justification J contained in the JER $\mathbf{J}$).

Justifications can have many uses for CBR systems: in a previous work [5] we applied justifications in order to improve the classification accuracy of the committee collaboration strategy. In this paper, we are going to use justifications to compute the expected utility of individual cases in order to create case retention strategies.

4 Collaborative Case Bargaining Retention Strategy

When an agent has access to a new case, a Case Retention strategy is needed to decide whether to incorporate this new case into the agent's case base or not. In this section we are going to present a collaborative retention strategy called *Cooperative Case Bargaining* (CCB).

The CCB strategy is a collaborative case retention strategy that tries to maximize the utility of the cases retained in the individual agents' cases bases. The basic idea of the CCB strategy is the following: each agent has an individual case utility estimation policy with which the agent can estimate the utility of retaining a given case (i.e. how much the new case will contribute to the agent's performance if retained). Moreover, different agents may assign different utility values to to same case, and a case that has a low utility for an agent may have a high utility for another agent. Using the CCB strategy, agents that receive cases with low utility values can give them to other agents if the case has a higher utility for them (expecting to be reciprocated in the future).

As all the collaboration strategies in $\mathcal{M}$AC systems, we will define the CCB strategy as an interaction protocol and a set of individual agent policies. In this section we will present the *Cooperative Case Bargaining* (CCB) protocol, and the *Justification-based Case Utility* (JCU) policy, an utility function based on justifications is used to estimate the utility of a given case for an agent.

4.1 Cooperative Case Bargaining Protocol

In this section, we are going to explain the Cooperative Case Bargaining (CCB) protocol, designed to perform case retention among a group of agents. The CCB protocol is based in four principles:

- Utility assessment: the individual agents are able to assess an utility value to estimate how much a new case will contribute to the agent's performance if retained in its individual case base.
- Delayed retention: when an agent A_i receives new cases, they are stored in a *pool of delayed retention cases* B_i instead of being retained directly into A_i's case base. These individual pools have a limited size m.
- Bargaining of cases: all the agents in the system compute the utility value for any case c_k in a pool of delayed retention cases, and then bargain for those cases with maximum utility for them.
- Small competent case bases: the protocol assumes that the goal of the agents is to achieve competent cases bases with a minimum number of cases necessary for a good performance to be maintained.

In the following we will first informally describe some aspects of the CCB protocol, and at the end of the section, the CCB protocol is formally presented.

When an agent A_i uses the CCB protocol all the new cases received go to B_i (the pool of A_i). When the pool B_i is full, A_i sends a message to the rest of agents stating that A_i wants to start the CCB protocol. The agent A_i that initiates the CCB protocol is called the *convener* agent. During the CCB protocol all the cases in the pools of the agents (including the cases in the pool of the convener agent A_i) will be bargained for by the agents.

Before the bargaining starts, every agent should notify the rest of agents about the contents of its local pool of delayed retention cases (so that every agent knows which are the cases that are going to be bargained). We will call $B = \bigcup_j B_j$ to the union of the pools of delayed retention cases.

The bargaining among the agents for the cases in B is performed in a series of rounds. At each round t an agent will retain a case. Therefore at each round t we can define the set $B^t \subseteq B$ as the set of cases that still haven't been retained by any agent. In the first round, $t = 0$ and $B^t = B$.

At each round t every agent A_j computes an *utility record* $\mathbf{U}$ for each case $c_k \in B^t$. An utility record $\mathbf{U} = \langle A, C, V \rangle$ is a record containing the utility value V computed by the agent A for the case C. For each case $c_k \in B^t$, an agent A_j will compute the utility record $\mathbf{U} = \langle A_j, c_k, u_j(c_k) \rangle$. We will note by $U_j = \{\langle A_j, c_k, u_j(c_k) \rangle | c_k \in B^t\}$ to the set of all the utility records computed by an agent A_j in a round t. When all the agents have computed these utility records, they are sent to the convener agent. In a second step, the convener agent examines all the utility records for each case in B^t, and selects the record $\mathbf{U}^t$ with the highest utility value. Finally, the agent $\mathbf{U}^t.A$ receives the case $\mathbf{U}^t.C$ and retains it. This finishes one round, and in the next round the agents will continue bargaining for the cases still not retained by any agent. The bargain ends when no agent is interested in any of the remaining cases (when an agent A_j sends an

utility equal to zero for a case c_k, we say that A_j is not interested in the case c_k) or when there are no more cases to bargain (i.e. $B^t = \emptyset$). When the bargaining ends because no agent is interested in any of the remaining cases, the cases in B^t are discarded (of course, the agents cannot be sure that the discarded cases will not become interesting in the future, but they are discarded to save space in the pools of cases, expecting to receive more interesting cases in the future).

Notice that when an agent A_j retains a case $c_k \in B$ the individual utility values of A_j must be recomputed (since the case base C_j of A_j has changed). Moreover, in the case of a tie (i.e. more than one agent have given the same maximum utility for some case), the winner is chosen randomly (but any other more informed criterion can be used). Notice also that in order to use the CCB protocol, the agents should have agreed before in some parameters of the protocol (such as the size of the pools, etc.) in the following, we will assume that all the agents have previously agreed in such parameters.

Specifically, the CCB protocol for a set of agents $\mathcal{A}$ is defined as follows:

1. An agent $A_i \in \mathcal{A}$ decides to initiate the CCB protocol because its pool of delayed retention cases B_i is full, and sends an initiating message to the rest of agents in $\mathcal{A}$. A_i will be called the *convener* agent.
2. The other agents in $\mathcal{A}$ send an acknowledgment message to the convener agent A_i meaning that they are ready to start the CCB protocol.
3. A_i broadcasts the cases contained in its pool B_i to the rest of agents.
4. In response to A_i, the rest of agents also broadcast the cases in their pools to the other agents.
5. When an agent A_j receives all the cases from the pools of the rest of agents, an acknowledgment message is sent back to A_i.
6. When A_i has received the acknowledgments from the rest of agents, every agent can compute the set $B = \bigcup_j B_j$. The first round $t = 0$ starts with $B^t = B$, and A_i broadcasts a message requesting for the individual utility records.
7. Every agent A_j computes the set of utility records $U_j = \{\langle A_j, c_k, u_j(c_k)\rangle | c_k \in B^t\}$ (computed using its own utility function. In our experiments, using the JCU policy), and sends them to the convener agent.
8. When A_i has received the utility records U_j from every agent A_j (and has computed its own utility values U_i), the record with the highest utility $\mathbf{U}^t \in \bigcup_j U_j$ is selected.
   - If $\mathbf{U}^t.V > 0$, A_i sends the case $\mathbf{U}^t.C$ to $\mathbf{U}^t.A$, and also sends a message to the rest of agents telling the the agent $\mathbf{U}^t.A$ has received the case $\mathbf{U}^t.C$. The protocol moves to state 9.
   - Otherwise ($\mathbf{U}^t.V = 0$), A_i sends a message to the rest of agents telling that the protocol is over and the remaining cases are discarded. The protocol ends.
9. All the agents send a message to A_i acknowledging that the round is over.
10. A_i computes the set of remaining cases for the next round $B^{t+1} = B^t - \{\mathbf{U}^t.C\}$. If $B^{t+1} \neq \emptyset$, A_i sends a message to the rest of agents requesting their new utility records for the remaining cases in B^{t+1}. A new round $t+1$

starts and the protocol moves to state 7. Otherwise, A_i sends a message to the rest of agents telling that the protocol is over. The protocol ends.

Delayed retention allows the agents to have a pool of cases to compute the utility from (using the JCU policy), and bargaining cases ensures that a good distribution of cases among the agents is achieved. The next section explains the Justification-based Case Utility policy used to assess case utility.

4.2 Justification-based Case Utility Policy

The *Justification-based Case Utility* (JCU) policy uses justifications in order to estimate the utility of adding a case c_k to the case base of an agent. The basic idea of the JCU policy is to determine if a case c_k will prevent a CBR agent to perform classification errors in the future. Therefore, the JCU policy favors cases that increase the classification accuracy of the system without taking into account the size of the case base.

Let agent A_j have access to a set of cases $B = \{(P_1, S_{P_1}), ..., (P_m, S_{P_m})\}$. None of cases in B is present in the agent's case base, therefore they are all candidates to be retained. However, before retaining any case, A_j wants to compute an utility function to decide which of them are worth retaining in the case base. For this purpose we define the set $E = \{P_j | (P_j, S_{P_j}) \in B\}$ as the set of all the problems contained in the cases in B.

To estimate the case utility values using the JCU policy, an agent A_j has to individually solve each one of the problems in E. After the agent has solved each problem in E, a *justified endorsing record* is build for each case. We will note $\mathbf{J}_E = \{\mathbf{J} | \mathbf{J}.P \in E\}$ as the set of JERs build by A_j for all the problems in the set E. Notice that the agent knows the correct solution for each of those problems, therefore the agent can test for each individual problem $P \in E$ whether P has been solved correctly or not. Thus, the agent can define $\mathbf{J}_E^- = \{\mathbf{J} | \mathbf{J} \in \mathbf{J}_E \wedge \mathbf{J}.S \neq S_{\mathbf{J}.P}\}$ as the set of JERs of the problems in E that A_j has solved incorrectly (where $S_{\mathbf{J}.P}$ is the correct solution class for the problem $\mathbf{J}.P$).

We can say that a case $c_k = (P_k, S_k)$ is a *counterexample* of an incorrect JER $\mathbf{J} \in \mathbf{J}_E^-$ if c_k is subsumed by the incorrect justification $\mathbf{J}$ and c_k belongs to a different solution class than the predicted one, i.e. $\mathbf{J}.J \sqsubseteq P_k$ and $S_k \neq \mathbf{J}.S$. Moreover, we can define also a *valid counterexample* of an incorrect justification as a counterexample c_k that belongs to the correct solution class of the problem P for which the justification $\mathbf{J}$ was created. i.e. a counterexample such that $S_k = S_{\mathbf{J}.P}$. Notice that the condition $S_k = S_{\mathbf{J}.P}$ implies that $S_k \neq \mathbf{J}.S$ if $\mathbf{J}$ is an incorrect JER. With the notion of valid counterexample, we can define the *refutation set*:

Definition: The *refutation set* $R_{\mathbf{J}}^B$ drawn from a pool of cases B for an incorrect JER $\mathbf{J}$ is defined as the set of cases from B that are valid counterexamples of that JER. Formally: $R_{\mathbf{J}}^B = \{(P_k, S_{P_k}) \in B | \mathbf{J}.J \sqsubseteq P_k \wedge S_{P_k} = S_{\mathbf{J}.P}\}$.

Notice that the cases in a refutation set $R_{\mathbf{J}}^B$ are the cases from B that can potentially prevent A_j from making the same error in the future (since they are valid counterexamples of the justification provided by A_j). We will call

$\mathcal{R} = \{R^B_{\mathbf{J}} | \mathbf{J} \in \mathbf{J}^-_{B^p}\}$ the collection of all the refutation sets for all the incorrect justifications $\mathbf{J}^-_E$.

We can now define the *utility* $u_i(c_k)$ of a case c_k in terms of the number of errors that it will fix for an agent A_j. If a case $c_k \in B$ is not present in any refutation set in $\mathcal{R}$, that case cannot fix any of the errors made by A_j while solving the problems in E. However, if a case $c_k \in B$ is present in some of the refutation sets in $\mathcal{R}$, c_k can fix some of the errors made by the agent. We will use the number of refutation sets $R^B_{\mathbf{J}}$ where a case c_k is present as as utility measure, that will be called *Justification-based Case Utility* (JCU):

$$u_i(c_k) = \#(\{R^B_{\mathbf{J}} \in \mathcal{R} | c_k \in R^B_{\mathbf{J}}\})$$

Notice that the utility estimation for a case c_k depends on two factors: the case base C_i of the agent (the better the case base is, the less the errors made in the set E, and the less the utility of new cases will be), and of the set of cases B. The larger (and more representative) the set B is, the more accurate the utility values assessment will be. This is the reason for delayed retention in the CCB protocol: the larger the agents' pools, the larger the set of cases B will be and the more accurate the utility values assessment will be.

In JCU, justifications help to identify which cases can help avoiding errors in solving the problems in E. Notice also that an utility equal to 0 means that an agent is not interested in that case.

We can summarize the process of determining the utility of a set of cases B for an agent A_j as follows:

1. Let $E = \{P_j | (P_j, S_{P_j}) \in B\}$ be the set with the problems in B.
2. Let $\mathbf{J}_E = \{\mathbf{J} | \mathbf{J}.P \in E\}$ be the set of JERs for the problems in E.
3. Let $\mathbf{J}^-_E = \{\mathbf{J} | \mathbf{J} \in \mathbf{J}_E \wedge \mathbf{J}.S \neq S_P\}$ be the set of incorrect JERs.
4. Let $\mathcal{R} = \{R^B_{\mathbf{J}} | \mathbf{J} \in \mathbf{J}^-_{B^p}\}$ be the collection of refutation sets.
5. Compute $u_i(c_k) = \#(\{R^B_{\mathbf{J}} \in \mathcal{R} | c_k \in R^B_{\mathbf{J}}\})$ for each $c_k \in B$.

The JCU values could be normalized between 0 and 1 dividing by the size of the set B, but for simplicity no normalization is applied. Next section presents an example of the execution of the BCC protocol and of the JCU policy.

4.3 Example

Let us illustrate the behavior of the CCB protocol with an example. Consider a system composed of 3 agents $\{A_1, A_2$ and $A_3\}$, that have individual pools of delayed retention cases B_1, B_2 and B_3 that can store 3 cases each. At a given time, the pools of the three agents contain the following cases: $B_1 = \{c_1, c_2, c_3\}$, $B_2 = \{c_4\}$ and $B_3 = \{c_5\}$, where $c_1 = (P_1, S_1)$, $c_2 = (P_2, S_2)$, etc.

When the pool B_1 of agent A_1 is full the agent A_1 initiates the CCB protocol. Both A_2 and A_3 broadcast the cases in their pools so that all the agents have access to the set of all delayed retention cases $B = \{c_1, c_2, c_3, c_4, c_5\}$.

When the first round $t = 0$ starts, all the agents apply the JCU policy to compute the utility records of the cases in $B^0 = B$. Let us focus on how agent

a) *Round 1*

	c_1	c_2	c_3	c_4	c_5
A_1	0	2	3	0	1
A_2	2	0	2	0	0
A_3	0	0	0	1	2

b) *Round 2*

	c_1	c_2	c_3	c_4	c_5
A_1	0	0	-	0	0
A_2	2	0	-	0	0
A_3	0	0	-	1	2

c) *Round 3*

	c_1	c_2	c_3	c_4	c_5
A_2	-	0	-	0	0
A_1	-	0	-	0	0
A_3	-	0	-	1	2

d) *Round 4*

	c_1	c_2	c_3	c_4	c_5
A_1	-	0	-	0	-
A_2	-	0	-	0	-
A_3	-	0	-	0	-

Table 1. Evolution of the utility values, for 3 agents A_1, A_2 and A_3 and a set $B = \{c_1, c_2, c_3, c_4, c_5\}$ of 5 cases in the CCB protocol.

A_1 uses the JCU policy: first, A_1 takes the set $E = \{P_1, ..., P_5\}$ and builds a JER for each problem in E. Assume that A_1 fails to correctly solve three problems, P_2, P_3 and P_5, and therefore the set $\mathbf{J}_E^- = \{\mathbf{J}_2, \mathbf{J}_3, \mathbf{J}_5\}$ has three JERs. A_1 builds then the refutation sets for those three JERs: $R^B_{\mathbf{J}_2} = \{c_2, c_3\}$, $R^B_{\mathbf{J}_3} = \{c_3\}$ and $R^B_{\mathbf{J}_5} = \{c_2, c_3, c_5\}$. With these refutation sets $\mathcal{R} = \{R^B_{\mathbf{J}_2}, R^B_{\mathbf{J}_3}, R^B_{\mathbf{J}_5}\}$ the JCU value of the 5 cases in B for the agent A_1 can now be computed:

$u^1(c_1) = \#(\emptyset) = 0$
$u^1(c_2) = \#(\{R^B_{\mathbf{J}_2}, R^B_{\mathbf{J}_5}\}) = 2$
$u^1(c_3) = \#(\{R^B_{\mathbf{J}_2}, R^B_{\mathbf{J}_3}, R^B_{\mathbf{J}_5}\}) = 3$
$u^1(c_4) = \#(\emptyset) = 0$
$u^1(c_5) = \#(\{R^B_{\mathbf{J}_5}\}) = 1$

In the same way, A_2 and A_3 compute their JCU values. All the agents send their utility records to A_1, that can now examine the utility records to determine the winner. Table 1.a shows the utility values for all the agents: the winner is the agent A_1, since the utility $u_1(c_3)$ is the highest. Therefore, A_1 retains the case c_3, the case is not available any more, and the rest of agents are notified.

When A_2 and A_3 answer with an acknowledgment to A_1, A_1 sends again a message to A_2 and A_3 requesting for the utility records of the remaining cases $B^1 = \{c_1, c_2, c_4, c_5\}$ for the second round of the protocol. A_1 has to recompute its own JCU values since has retained a new case, and the new JCU values are shown in Table 1.b. This time there is a tie between A_2 and A_3 that is resolved randomly: the winner is A_2, that receives the case c_1 to be retained.

The JCU values in the third round for the cases $B^2 = \{c_2, c_4, c_5\}$ can be seen in Table 1.c, where the winner is A_3 that receives the case c_5.

In the fourth round, no agent wants any case in $B^3 = \{c_2, c_4\}$, as shown in Table 1.d where all the JCU values are zero. A_1 sends a message to A_2 and A_3 telling that the CCB protocol is over, the cases c_2 and c_4 are discarded, and the pools of the three agents are cleared.

One may think that, if every agent has access to all the cases during the CCB protocol, why isn't it the best policy to allow each agent to retain every case?

In fact, allowing each agent to retain every case is not the best policy (as we are going to show in the experiments section), since the resulting system would be equivalent to a single agent (since as each agent would have all the cases). In the experiments section we will show how a group of agents using the CCB protocol can outperform a single agent that has all the cases.

The CCB protocol may appear to be a complex way to distribute the cases among the agents. However, it is designed in this way since the order in which the cases are bargained does matter. A simpler protocol that would consider the cases one at a time could lead to suboptimal results.

5 Experimental results

This section evaluates the performance of the CCB strategy and of the JCU policy. For that purpose, we are going to compare the performance of groups of agents using the CCB strategy against groups of agents using other retention strategies. The presented results will be related to classification accuracy and case base sizes of the agents.

We use the marine sponge classification problem as our test bed. We have designed an experimental suite with a case base of 280 marine sponges pertaining to three different orders of the *Demospongiae* class (*Astrophorida*, *Hadromerida* and *Axinellida*). The goal of the agents is to identify the correct biological order given the description of a new sponge. In each experimental run the whole collection of cases is divided in two sets, a training set (containing a 90% of the cases), and a test set (containing a 10% of the cases). The cases in the training set are sent to the agents incrementally, i.e. each problem in the training set arrives randomly to one agent in the $\mathcal{M}$AC. The agent receiving the case will apply a retention strategy to decide whether to retain the case or not. Each time that a 10% of the training set is sent to the agents, the test set is also sent to them to evaluate their classification accuracy at that moment in time. Thus, the test set is sent to the agents 11 times (one at the beginning, and 10 as each 10% of the training set is sent) to obtain the evolution of the classification accuracy of the agents as they retain cases from the training set. Both, the accuracy of the committee and the individual accuracy of the agents will be measured. The results presented in this section are the average of the accuracies obtained for the test sets in 5 10-fold cross validation runs. All the agents use the LID CBR method to solve problems.

These experiments evaluate the effectiveness of the collaborative learning policies, so it is important that the agents really have an incentive to collaborate. If every agent receives a representative (not biased) sample of the data, they will have a lower incentive to ask for cases to other agents since they already have a good sample. For this reason, for experimentation purposes, the agents do not receive the problems randomly. We force biased case bases in every agent by increasing the probability of each agent to receive cases of some classes and decreasing the probability to receive cases of some other classes. Therefore, each agent will have a biased view of the data. This will lead to a poor individual

performance, as we will see when we present the individual accuracy results, and an incentive to collaborate. However, we will also give some experimental results on the non biased scenario.

We will compare the performance of 4 different retention strategies:

- Cooperative Case Bargaining (CCB) strategy: This is the strategy presented in this paper, where the agents store cases in their pools of delayed retention cases, and when the pools are full the CCB protocol is engaged.
- Always Retain (AR) strategy: In this strategy, an agent simply retains all the cases individually received.
- On Failure Retain (OFR) strategy: In this strategy, an agent tries to solve a case before retaining it. If the agent fails to solve the problem correctly, the case is retained (this strategy is essentially that of IB2 [1] for instance based learners).
- Individual Justification Case Utility (IJCU) strategy: In this strategy, the agents store cases in their pools, but when the pools are full, the agents simply apply the JCU policy to decide which cases to retain from their individual pool without sharing cases with other agents. The cases not wanted by the agents are discarded. Since this strategy avoids collaboration, it is a valid retention strategy for individual CBR systems as well.

Figure 2 shows the evolution of the classification accuracy for a 5 agents $\mathcal{M}$AC where the agents use the committee collaboration strategy to solve problems. Four different lines are shown, one per each retention strategy tested. The horizontal axis shows the percentage of cases from the training set that the agents have received. Notice that each problem of the training set is only received by a single agent. Therefore, each agent receives only a fifth of the total training set (since there are 5 agents in our experiments).

Figure 2 shows that the agents using the CCB retention strategy obtain the highest accuracies, reaching an accuracy of 91.5%. Agents using the AR (retaining every case they receive) are behind the agents using CCB, reaching an accuracy of 88.14%. Finally, OFR and IJCU reach similar accuracies: 83.78% and 84.57% respectively, but with IJCU winning for a slight difference. Notice that the IJCU curve grows a little slower at the beginning than OFR (due to the delayed retention), but this effect is very soon compensated.

For comparison purposes, notice that the classification accuracy of a single agent owning *all* the cases in the training set (i.e. the accuracy of a centralized approach) is of 88.20% (lower than the accuracy of 91.5% obtained by the committee using the CCB strategy). Moreover, the accuracy obtained by the committee using a unbiased distribution of cases among the 5 agents (i.e. using AR without bias) is of 88.36%, still lower than the accuracy obtained by CCB. Therefore, we can conclude that CCB obtains a better distribution of cases than a random unbiased distribution or a centralized approach.

We can also compare the different retention strategies concerning the sizes of the case bases of the CBR agents at the end of the experiments shown in Table 2. The strategy that obtains larger case bases is the AR strategy (since

Fig. 2. Comparison of the evolution of classification accuracy for the committee using different retention strategies.

CCB	AR	OFR	IJCU
27.6	50.4	19.0	16.6

Table 2. Average case base sizes of the individual agents after having received all the cases o the training set.

the agents always retain every case they receive), with an average size of 50.4 cases. Agents using the CCB strategy retain only 27.6 cases in average, about a 55% of the cases retained by the AR strategy. The OFR strategy retains less cases, obtaining an average of 19.0 cases per agent. Finally, the strategy that obtains smaller case bases is the IJCU strategy (where the agents use the IJUC policy, but only with the cases in their local pools).

Taking into account both the results in classification accuracy and the case base size we can conclude that CCB is clearly better than AR, since retains less cases and achieves higher accuracies. IJCU is also clearly better than OFR, since IJCU achieves slightly higher accuracies and with smaller case bases. We can also see that CCB is clearly better than IJCU and OFR, since there is a large increase on accuracy of CCB with respect to IJCU and OFR. The cases that CCB retains (and OFR or IJCU do not) are the reason for the increased classification accuracy. Moreover, since CCB is equivalent to adding collaboration to IJCU, we can conclude that collaboration is beneficial for the CBR agents.

Finally, we also present results of individual classification accuracy. Figure 3 shows the evolution of classification accuracy for a 5 agents $\mathcal{MAC}$ where the agents solve problems individually. This time, the increment in classification accuracy of CCB with respect to the rest of strategies is increased: agents using CCB obtain a 82.88% of individual accuracy, agents using AR obtain a 73.11% of classification accuracy, agents using OFR a 66.34% and agents using IJCU a

Fig. 3. Comparison of the evolution of agents' individual classification accuracy using different retention strategies.

66.88%. The increase in the committee accuracy obtained by CCB with respect to the other retention strategies is mostly due to the increase of accuracy of the individual CBR agents, from 73.11% to 82.88%. Moreover, the increase from the individual accuracy to the committee accuracy is due to the ensemble effect, from 82.88% to 91.5% with the CCB strategy. Notice that the ensemble effect requires that the errors made by the individual agents are not correlated. Therefore, we can conclude that the CCB strategy is able to keep the error correlation among the agents low so that they can still benefit from the ensemble effect.

6 Conclusions

This paper has presented the Collaborative Case Bargaining (CCB) strategy for case retention, in which individual agents collaborate in order to select which cases to retain. We have introduced the concept of justification. A justification contains information concerning why a CBR agent has classified a problem into a specific solution class. We have presented the Justification-based Case Utility (JCU) policy, that is able to compute an utility measure of the cases to be retained using justifications. Justifications allow JCU to determine which cases can avoid making errors in the future, and give higher utility values to those cases that can avoid the higher number of errors in the future. Therefore, justifications have proven to be a useful tool in CBR systems. We have also shown that using the CCB strategy in combination with the committee collaboration strategy, agents can obtain better results than a centralized approach (where a single case base would contain all the available cases).

Moreover, we have introduced the concept of delayed retention. By using delayed retention, a CBR agent does not decide whether to retain a case until a pool of delayed retention cases is full. This allows the CBR agents to better

decide which are the cases to retain in the case base. Moreover, delayed retention requires a certain amount of cases to perform good estimation of the utility of the cases. In our experiments, the size of the individual agents' pools is 5. However, we plan to make experiments with different pool sizes. A larger pool size implies a better utility assessment, but at the cost of delaying the learning process, so some tradeoff is needed. Notice also that delayed retention and the JCU policy can be also applied to centralized CBR systems, since no collaboration is needed. For instance, a CBR system with all the cases using the JCU policy has an accuracy of 86.42% while retaining only 61.1 cases in average (a 24.24% of the total number of cases).

The distribution of retained cases among the CBR agents plays a main role in the final classification accuracy obtained by the committee of CBR agents. We have seen that the CCB strategy allows the CBR agents to obtain good distributions of cases. As a future work we plan to develop further strategies to improve the distribution of cases among CBR agents. CCB selects good cases for retention, but once an agent has retained a case, no other agent can retain it, and it will be never moved to another agent's case base. We plan to develop strategies for case redistribution among individual agents' case bases to improve the individual and collective performance.

Acknowledgements The authors thank Eva Armengol and Josep-Lluís Arcos of the IIIA-CSIC for the development of the LID and of the Noos agent platform respectively. Support for this work came from CIRIT FI/FAP 2001 grant and project SAMAP (MCYT-FEDER) TIC2002-04146-C05-01.

References

[1] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[2] E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In Luc de Raedt and Peter Flach, editors, *EMCL 2001*, number 2167 in Lecture Notes in Artificial Intelligence, pages 13–24. Springer-Verlag, 2001.

[3] L. K. Hansen and P. Salamon. Neural networks ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001, 1990.

[4] S. Ontañón and E. Plaza. Cooperative case retention strategies for cbr agents. In Derek Bridge and Kevin Ashley, editors, *ICCBR-2003*. Springer-Verlag, 2003.

[5] S. Ontañón and E. Plaza. Justification-based multiagent learning. In *Proc. 20th ICML*, pages 576–583. Morgan Kaufmann, 2003.

[6] E. Plaza and S. Ontañón. Ensemble case-based reasoning: Collaboration policies for multiagent cooperative cbr. In I. Watson and Q. Yang, editors, *ICCBR-2001*, number 2080 in LNAI, pages 437–451. Springer-Verlag, 2001.

[7] B. Smyth. The utility problem analysed: A case-based reasoning persepctive. In *EWCBR-96*, LNAI, pages 234–248. Springer Verlag, 1996.

[8] Frank van Harmelen. How the semantic web will change KR. *The Knowledge Engineering Review*, 17(1):93–96, 2002.

[9] Bruce A. Wooley. Explanation component of software systems. *ACM Crossroads*, 1998.