Ensemble Case-based Reasoning: Collaboration Policies for Multiagent Cooperative CBR

Enric Plaza and Santiago Ontañón

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

Abstract. Multiagent systems offer a new paradigm to organize AI applications. Our goal is to develop techniques to integrate CBR into applications that are developed as multiagent systems. CBR offers the multiagent systems paradigm the capability of autonomously learning from experience. In this paper we present a framework for collaboration among agents that use CBR and some experiments illustrating the framework. We focus on three collaboration policies for CBR agents: Peer Counsel, Bounded Counsel and Committee policies. The experiments show that the CBR agents improve their individual performance collaborating with other agents without compromising the privacy of their own cases. We analyze the three policies concerning accuracy, cost, and robustness with respect to number of agents and case base size.

1 Introduction

Multiagent systems offer a new paradigm to organize AI applications. Our goal is to develop techniques to integrate CBR into applications that are developed as multiagent systems. Learning is a capability that together with autonomy is always defined as a feature needed for full-fledged agents. CBR offers the multiagent systems paradigm the capability of autonomously learning from experience. In this paper we present a framework for collaboration among agents that use CBR and some experiments illustrating the framework.

A distributed approach for CBR makes sense in different scenarios. Our purpose in this paper is to present a multiagent system approach for distributed case bases that can support these different scenarios. A first scenario is one where cases themselves are owned by different partners or organizations. This organizations can consider their cases as assets and they may not be willing to give them to a centralized "case repository" where CBR can be used. In our approach each organization keeps their private cases while providing a CBR agent that works with them. Moreover, the agents can collaborate with other agents if they keep the case privacy intact an they can improve their performance by cooperating. Another scenario involves scalability: it might be impractical to have a centralized case base when the data is too big.

Our research focuses on the scenario of separate case bases that we want to use in a decentralized fashion by means of a multiagent system, that is to say a collection of CBR agents that manage individual case bases and can communicate (and collaborate) with other CBR agents. In this paper we focus on three collaboration policies that improve the individual performance of CBR agents without compromising the agent's autonomy and the privacy of the case bases. These collaboration policies are a refinement of the general multiagent scenario of *Cooperative CBR* proposed in [9]. Particularly, CoopCBR established two cooperation modes, namely DistCBR and ColCBR[1]. The collaboration policies presented here are strategies that CBR agents can follow to improve their individual performance in the framework of the DistCBR cooperation mode.

The structure of the paper is as follows. Section 2 presents three policies the CBR agents can follow to improve their performance cooperating with other agents in a multiagent system. Then, section 3 presents the CBR method that the agents use in our current experiments. The experiments themselves are explained in section 4. The paper closes with related work and conclusion sections.

2 Policies for Cooperative CBR

A multiagent CBR ($\mathcal{M}$AC) system $\mathcal{M} = \{(A_i, C_i)\}_{i=1...n}$ is composed of n agents, where each agent A_i has a case base C_i. In the experiments reported here we assume the case bases are disjunct ($\forall A_i, A_j \in \mathcal{M}\mathsf{AC} : C_i \cap C_j = \emptyset$), i.e. there is no case shared by two agent's case bases. This is just an experimental option and not a restriction on our model. 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 $C_i = \{(P_j, S_k)\}_{j=1...N}$ is a collection of pairs problem/solution.

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 that is either `:sorry` (if it cannot solve P) or a solution endorsement record (SER). A SER has the form $\langle\{(S_k, E_k^j)\}, P, A_j\rangle$, 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.

Before presenting the three policies for cooperative CBR, *Committee*, *Peer Counsel* and *Bounded Counsel* policies, we will introduce the voting mechanism.

[1] Summarily, in the DistCBR mode each CBR agent uses its own similarity assessment to retrieve cases while in the ColCBR cooperation mode an agent sends to other agents the *method* to assess the similarity in the process of case retrieval.

2.1 Voting Scheme

The voting scheme defines the mechanism by which an agent reaches an aggregate solution from a collection of SERs coming from other agents. 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 do not want that agents having more number of endorsing cases may have an unbounded number of votes regardless of the votes of the other agents. 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 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 or equal than 1. Aggregating the votes from different agents for a class S_k we have ballot

$$Ballot^t(S_k, \mathcal{A}^t) = \sum_{A_j \in \mathcal{A}^t} Vote(S_k, A_j)$$

and therefore the winning solution class is

$$Sol^t(P, \mathcal{A}^t) = arg \max_{k=1\ldots K} Ballot(S_k, \mathcal{A}^t)$$

i.e., the class with more votes in total. We will show now three collaboration policies that use this voting scheme.

2.2 Committee Policy

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

The next two policies, *Peer Counsel* and *Bounded Counsel*, are based on the notion that an agent A_i tries to solve a problem P by himself and if A_i "fails" to find a "good" solution then A_i asks counsel to other agents in the $\mathcal{M}$AC system $\mathcal{M}$. Let $E_P^i = \{(S_k, E_k^i)\}$ the endorsement pairs the agent A_i computes to solve problem P. For an agent A_i to decide when it "fails" we require that each agent in $\mathcal{M}$ has a predicate *Self-competent*(P, E_P^i). This predicate determines whether or not the solutions endorsed in E_P^i allow the agent to conclude that there is a good enough solution for P.

2.3 Peer Counsel Policy.

In this policy the agents member of a $\mathcal{M}$AC system $\mathcal{M}$ try first to solve the problems they receive by themselves. Thus, if agent A_i receives a problem P and finds a solution that is satisfactory according to its own *Self-competent* predicate, the solution found is the final solution. However, if an agent A_i assesses that it is not capable of finding a reliable solution, then it asks the other agents in $\mathcal{M}$ to also solve the problem P.

The agents in $\mathcal{M}$ return to A_i their solution(s) inside their solution endorsement records and (as done in the committee policy) the final solution is the class with maximum number of votes.

2.4 Bounded Counsel Policy

In this policy the agents member of a $\mathcal{M}$AC system $\mathcal{M}$ try first to solve the problems they receive by themselves, as in the previous *Peer Counsel* policy. However, when an agent A_i assesses that its own solution is not reliable, the *Bounded Counsel* Policy tries to minimize the number of questions asked to other agents in $\mathcal{M}$. Specifically, agent A_i asks counsel only to one agent, say agent A_j. When the answer of A_j arrives the agent A_i implements a termination check. If the termination check is true the result of the voting scheme is the global result, otherwise A_i asks counsel to another agent—if there is one left to ask, if not the process terminates and the voting scheme determines the global solution.

The termination check works, at any point in time t of the *Bounded Counsel* Policy process, upon the collection of solution endorsement records (SER) received by the initiating agent A_i at time t. Using the same voting scheme as before, Agent A_i has at any point in time t a plausible solution given by the winner class of the votes cast so far. Let V^t_{max} be the votes cast for the current plausible solution, $V^t_{max} = Ballot^t(Sol^t(P, \mathcal{A}^t), \mathcal{A}^t)$, the termination check $TC(V^t_{max}, \mathcal{A}^t)$ is a boolean function that determines whether there is enough difference between the majority votes and the rest to stop and obtain a final solution. In the experiments reported here the termination check function (applied when there are votes for more than one solution class) is the following

$$TC(V^t_{max}, \mathcal{A}^t) = \frac{V^t_{max}}{\left(\sum_{S_k \in K} Ballot(S_k, \mathcal{A}^t)\right) - V^t_{max}} \geq \eta$$

i.e. it checks whether the majority vote V^t_{max} is η times bigger than the rest of the ballots. After termination the global solution is the class with maximum number of votes at that time.

Policy Varieties There may be small variations on this policies that follow the same overall schema. For instance, we are assuming that the global solution implies selecting a single alternative $S_k \in K$. However, depending on the task at hand, a ranked list of possible solutions might be a better option. We can easily

adapt the present policies for tasks with ranked solutions: the final step in voting that takes the solution with maximum number of votes can be substituted by yielding k highest solutions ranked by their respective number of votes. Nevertheless, for the experiments reported later, it is more convenient for comparisons purposes to work with the hypothesis that a single solution is required.

We call this framework *Ensemble CBR* since a meaning of *ensemble* (Oxford Dictionary) is this: "*(Math)* a group of systems with the same constitution but possibly with different states". From the point of view of the $\mathcal{M}$AC framework, a system $\mathcal{M} = \{(A_i, C_i)\}_{i=1\ldots n}$ performs Ensemble CBR when the CBR agents $A_1 \ldots A_n$ work with the CBR method but they have different experience (different case bases $C_1 \ldots C_n$).

The collaboration policies described here have been implemented on the Noos Agent Platform [8]. NAP consists of Noos, a representation language with support for case management and retrieval [1], and FIPA-compliant utilities for agent interaction. A multiagent system in NAP consists of the individual agents capabilities (like CBR) plus a specification of the agent roles and interaction protocols in the framework of agent-mediated institutions [8]. Cases are represented as feature terms in Noos and the next section introduces the CBR method used in our CBR agents.

3 Case-based Reasoning Agents

In the following section we will introduce the concepts needed to explain LID [2], the CBR method used by the agents. First we will introduce feature terms, the representation used for cases; then we will explain the heuristic measure used by LID, and finally the LID algorithm will be described.

3.1 Representation of the cases

LID handles cases represented as feature terms. *Feature Terms* (also called feature structures or ψ-terms) are a generalization of first order terms. The difference between feature terms and first order terms is the following: a first order term, e.g. $f(x, y, g(x, y))$ can be formally described as a tree and a fixed tree-traversal order. In other words, parameters are identified by position. The intuition behind a feature term is that it can be described as a labelled graph i.e. parameters are identified by name. A formal definition of feature terms is the following:

Given a signature $\Sigma = \langle S, \mathcal{F}, \leq \rangle$ (where S is a set of sort symbols that includes $\bot$; $\mathcal{F}$ is a set of feature symbols; and $\leq$ is a decidable partial order on S such that $\bot$ is the least element) and a set ϑ of variables, we define *feature terms* as an expression of the form:

$$\psi ::= X : s[f_1 \doteq \Psi_1 \ldots f_n \doteq \Psi_n] \tag{1}$$

where X is a variable in ϑ called the *root* of the feature term, s is a sort in S, the function *root(ψ)* returns the sort of the root, $f_1 \ldots f_n$ are features in $\mathcal{F}$,

$n \geq 0$, and each Ψ_i is a set of feature terms and variables. When $n = 0$ we are defining a variable without features. The set of variables occurring in ψ is noted as ϑ_ψ.

Sorts have an informational order relation ($\leq$) among them, where $\psi \leq \psi'$ means that ψ has less information than ψ' or equivalently that ψ is more general than ψ'. The minimal element ($\perp$) is called *any* and it represents the minimum information. When a feature has an unknown value it is represented as having the value *any*. All other sorts are more specific that *any*.

A *path* $\rho(X, f_i)$ is defined as a sequence of features going from the variable X to the feature f_i.

There is a *path equality* when two paths point to the same value. Path equality is equivalent to variable equality in first order terms.

The *depth* of a feature f in a feature term ψ with root X is the number of features that compose the path from the root X to f, including f, with no repeated nodes.

Given a particular maximum feature depth k, a *leaf feature* of a feature term is a feature f_i such that either 1) the depth of f_i is k or 2) the value of f_i is a term without features.

3.2 Heuristic assessment of feature relevance

The heuristic used in LID is the minimization of the RLM distance [5]. The RLM distance assesses how similar are two partitions over a set of cases (in the sense that the lesser the distance the more similar they are). On the one hand, we have the correct partition given by the classified cases in the case base $C_i = \{(P_j, S_k)\}_{j=1\ldots N_i}$ of agent A_i. Formally, the *correct partition* is the collection of sets $\Pi_C(C_i) = \{\pi_{S_k}\}_{k=1\ldots K}$ where $\pi_{S_k} = \{P_z | (P_z, S_k) \in C_i\}$. On the other hand, each feature f that has legal values $v_1 \ldots v_{nf}$ induces also a partition over the case base, namely a partition whose sets are formed by those cases that have the same value for feature f. Formally, the induced partition is the collection of sets $\Pi_f(C_i) = \{\pi^f_{v_j}\}_{j=1\ldots nf}$ where a set is $\pi^f_{v_j} = \{P_w | \exists S_k : (P_w, S_k) \in C_i \wedge P_w.f = v_j\}$.

In the following, we will use RLM distance over a variable case base B and not over the whole case base C_i. For a partition $\Pi_f(B)$ induced by a feature f, LID computes the RLM distance to the correct partition $\Pi_c(B)$. Given two partitions Π_f and Π_c of the case base B, the RLM distance between them is computed as follows:

$$RLM(\Pi_f, \Pi_c) = 2 - \frac{I(\Pi_f) + I(\Pi_c)}{I(\Pi_f \cap \Pi_c)}$$

where $I(\Pi_f)$ and $I(\Pi_c)$ measure the information contained in the partition Π_f and Π_c respectively and $I(\Pi_f \cap \Pi_c)$ is the mutual information of the two partitions.

Let Π_c be the correct partition and Π_f and $\Pi_{f'}$ the partitions induced by features f and f' respectively. We say that the feature f is *more discriminatory*

```
Function LID (S_D, P, D, K)
    if stopping-condition(S_D)
        then return SER(S_D, P)
        else f_d := Select-leaf (p, S_D, K )
             D' := Add-path(ρ(root(P), f_d), D)
             S_D' := Discriminatory-set (D, S_D)
             LID (S_D', P, D', C)
    end-if
end-function
```

Fig. 1. The LID algorithm. $\mathcal{D}$ is the similitude term, $\mathcal{S}_\mathcal{D}$ is the discriminatory set of $\mathcal{D}$, K is the set of solution classes, $SER(\mathcal{S}_\mathcal{D}, P)$ constructs the Solution Endorsement Record of problem P.

than the feature f' if $RLM(\Pi_f, \Pi_c) < RLM(\Pi_{f'}, \Pi_c)$. In other words, when a feature f is more discriminatory than another feature f' the partition that f induces in B is closer to the correct partition Π_c than the partition induced by f'. Intuitively, the most discriminatory feature classifies the cases in B in a more similar way to the correct classification of cases.

3.3 The LID method

The main steps of the LID algorithm are shown in Figure 1. Initially, LID of agent A_i receives as parameter $\mathcal{S}_\mathcal{D}$ the whole case base C_i and parameter $\mathcal{D}$ is empty.

The top down process of LID specializes the similitude term $\mathcal{D}$ by adding features to it. In principle, any of the features used to describe the cases could be a good candidate. Nevertheless, LID uses two biases to obtain the set F_l of features candidate to specialize the current similitude term $\mathcal{D}$. First, of all possible features in the domain $\mathcal{F}$, LID will consider only those features present in the problem P to be classified. As a consequence, any feature that is not present in P will not be considered as candidate to specialize $\mathcal{D}$. The second bias is to consider as candidates for specializing $\mathcal{D}$ only those features that are leaf features of P—i.e. a feature f_i such that either 1) the depth of f_i is equal to a depth threshold k or 2) the value of f_i is a term without features.

The next step of LID is the selection of a leaf feature $f_d \in F_l$ to specialize the similitude term $\mathcal{D}$. Let F_l be the set of leaf features candidates to specialize $\mathcal{D}$. Selecting the most discriminatory leaf feature in the set F_l is heuristically done using the RLM distance which is explained in section 3.2. Let us call the most discriminatory feature f_d.

The feature f_d is the leaf feature of path $\rho(root(P), f_d)$ in problem P. The specialization step of LID defines a new similitude term $\mathcal{D}'$ by adding to the current similitude term $\mathcal{D}$ the sequence of features specified by $\rho(root(P), f_d)$. After this addition $\mathcal{D}'$ has a new path $\rho(root(\mathcal{D}'), f_d)$ with all the features in the path taking the same value that they take in P. After adding the path ρ to $\mathcal{D}$,

the new similitude term $\mathcal{D}' = \mathcal{D} + \rho$ subsumes a subset of cases in $\mathcal{S}_\mathcal{D}$, namely the discriminatory set $\mathcal{S}_{\mathcal{D}'}$ (the subset of cases subsumed by $\mathcal{D}'$).

Next, LID is recursively called with the discriminatory set $\mathcal{S}_{\mathcal{D}'}$ and the similitude term $\mathcal{D}'$. The recursive call of LID has $\mathcal{S}_{\mathcal{D}'}$ as first parameter (instead of $\mathcal{S}_\mathcal{D}$) because the cases that are not subsumed by $\mathcal{D}'$ will not be subsumed by further specialization. The process of specialization reduces the discriminatory set $\mathcal{S}^n_\mathcal{D} \subseteq \mathcal{S}^{n-1}_\mathcal{D} \subseteq \ldots \subseteq \mathcal{S}^0_\mathcal{D}$ at each step.

The result of LID solving a problem P is a Solution Endorsement Record $\langle \{(S_k, E^j_k)\}, P, A_j \rangle$. When the termination condition is that all cases in the discriminatory set $\mathcal{S}_\mathcal{D}$ belong to only one solution S_k, the SER is simply $\langle (S_k, M_k), P, A_j \rangle$, where $M_k = |\mathcal{S}_\mathcal{D}|$. Otherwise, the SER is built in a similar way for each solution class in $\mathcal{S}_\mathcal{D}$ computing the number of cases endorsing each class $M_k = |\{(P_i, S_k) \in \mathcal{S}_\mathcal{D}\}|$

4 Experiments

We use the marine sponge identification (classification) problem as our testbed. Sponge classification is interesting because the difficulties arise from the morphological plasticity of the species, and from the incomplete knowledge of many of their biological and cytological features. Moreover, benthological specialists are distributed around the world and they have experienced different benthos that spawn species with different characteristics due to the local habitat conditions. Therefore the task of marine sponge identification is inherently distributed among agents that have a non-complete experience of their domain of expertise. We have chosen to follow this scenario in our experiments. specifically we use a sample of marine sponges that will be distributed among a number of agents' cases bases in such a way that they are disjoint (no specimen is in more than one case base).

This scenario is quite different from other multimodel classification approaches where different (classifier) agents have access to all the examples and propose candidate classifications that later are aggregated into a global solution (see §5).

In order to compare the performance of the three policies, 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.

We have experimented with 3, 4, 5, 6 and 7 agents using LID as its CBR method. The results presented here are the result of the average of 5 10-fold cross validation runs. Therefore, as we have 280 sponges in our case base, in each run 252 sponges will form the training set and 28 will form the test set.

In an experimental run, training cases are randomly distributed to the agents (without repetitions, i.e. each case will belong to only one agent case base). Thus, if we have n agents and m examples in the training set, each agent should have about m/n exampes in its case base. Therefore increasing the number of agents

Policy	*3 Agents*		*4 Agents*		*5 Agents*		*6 Agents*		*7 Agents*	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Isolated Agents	83.21	6.71	82.50	6.44	79.43	8.44	77.93	7.55	75.78	6.82
Bounded Counsel	87.29	6.1	86.71	6.47	85.07	6.29	85.00	7.25	84.14	7.04
Peer Counsel	87.28	5.72	86.79	6.67	85.85	6.68	85.50	5.86	84.71	6.75
Committee	88.36	5.98	88.29	5.72	88.36	5.41	88.14	6.04	87.93	5.86

Table 1. Average precision and standard deviation for a case base of 280 sponges pertaining to three classes. All the results are obtained using a 10-fold cross validation.

in our experiments their case-base size decreases. When all the examples in the training set have been distributed, the test phase starts.

In the test phase, for each problem P in the test set, we randomly choose an agent A_i and send P to A_i. Thus, every agent will only solve a subset of the whole test set. If testing the isolated agents scenario, A_i will solve the problem by itself without help of the other agents. And if testing any of the collaboraton policies, A_i will send P to some other agents.

4.1 Experimental Results

We can see (Table 1) that in all the cases we obtain some gain in accuracy compared to the isolated agents scenario. The Committee policy is always better than the others; however this precision has a higher cost since a problem is always solved by every agent. We evaluate costs later in this section. If we look at the cheaper policies *Bounded Counsel* and *Peer Counsel*, we can see that their accuracy are very similar. They both are much better than the isolated agents, and slightly worse than the Committee policy.

A small detriment of the system's performance is observable when we increase the number of agents. This is due to the fact that the agents have a more reduced number of training cases in their case bases. A smaller case base has the effect of obtaining less reliable individual solutions. However, the global effect of reducing accuracy appears on *Bounded Counsel* and *Peer Counsel* policies but not on the Committee policy. Thus, the Committee policy is quite robust to the effect of diminishing reliability individual solutions due to smaller case bases. This result is reasonable since the Committee policy always uses the information available from all agents.

In order to assess that *Committee*, *Peer Counsel* and *Bounded Counsel* policies obtain better accuracies that the isolated agents scenario, we have tested the statistical significance of the gains achieved by the collaboration policies. We have used the *signed rank* test with a confidence of 99% — meaning that we can be sure of the results with a confidence of the 99%. This test tells us when a method obtains numeric accuracies higher than another, and thus we can use it to compare the accuracies obtained by our collaboration policies. The results obtained are that all the collaboration policies are always significantly better than the isolated agents. *Committee* policy is always significantly better than

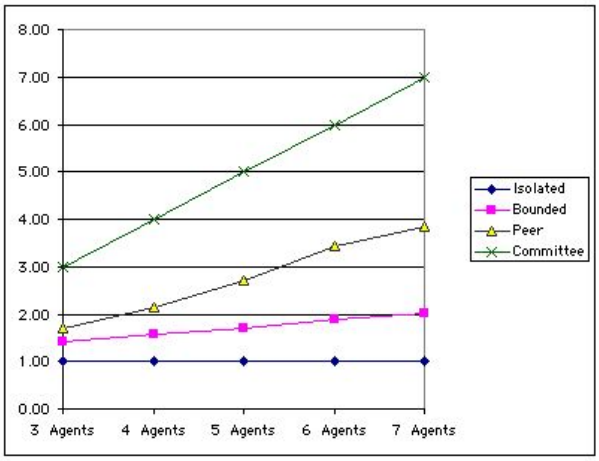

Fig. 2. Average cost (in euros) of solving a problem depending on the policy and number of agents.

Peer Counsel and *Bounded Counsel*. And the diferences between *Peer Counsel* and *Bounded Counsel* are not statistically significant.

A more realistic scenario in multiagent systems is to take into account the cost of cooperation among agents. Assuming that solving a problem has a cost, we have made some experiments where the nominal price for an agent solving a problem is 1 euro. Thus, an agent solving a problem by itself will have a cost of 1 euro—but if it asks two other agents to help the cost will be 3 euro. Having a cost on questions allows us to see how the different policies minimize the global cost and its tradeoff with accuracy. Figure 2 shows the average cost per problem in different policies and with a $\mathcal{M}$AC composed of 3 to 7 agents. The cost is calculated over the same 5 10-fold cross validation runs. In the isolated agents scenario the cost for solving a problem is always 1 euro because only one agent is involved per problem. On the other side we have the Committee policy, where all the agents are asked to solve each problem; in this policy the cost increases linearly with the number of agents, i.e.: if we have 5 agents the cost is 5 euros. With the *Peer Counsel* policy the cost per problem is lower and increases much more slowly than in the Committee policy— with only a small detriment of the accuracy. Finally, the cost of *Bounded Counsel* policy is lower than the *Peer Counsel* policy —and also increases much slowly. This outcome is as expected since Bounded Counsel tries to minimize questions asked to other agents. Moreover, since the accuracy of *Peer Counsel* is not statistically different than *Bounded Counsel*, we should prefer *Bounded Counsel* for having a lower cost.

Related to the cost in euros we have the computational cost of solving a problem. Figure 3 shows the cost in computation time, specifically it shows the average time spent to solve a problem when all the agents are running in the

same machine. We can see that there is a direct relation between the computational cost (time) and the cost in euros. As before the cheapest policy is *Bounded Counsel* (except for the isolated agents scenario) and the most expensive is the Committee policy. Now, observing Figure 3 it is apparent that having more agents reduces the time needed to compute a solution. This occurs in all policies except for the Committee policy where it's constant. The computation time has two components: the average retrieval time (in an agent) and the number of retrieval processes performed to solve a problem (the number of agents involved in solving a problem). The general trend in the reduction of computational time is due to the first component: the bigger the number of agents, the smaller the number of cases in a case base and, thus, the faster the retrieval process implemented by LID. On the other side, the number of retrieval processes involved can vary with the policy and the number of agents involved.

For instance, if we look with some detail the Figure 3 we may notice that *Peer Counsel* first starts decreasing and in the 6 and 7 agents columns it slightly increases again. If we look at the policy with detail, we can see that it's not so surprising. The fact is that when an agent has very few cases in the case base then more often its self-competence assessment indicates that the individual results are not reliable enough and has to ask counsel to the other peer agents. This effect is particularly noticeable in the *Peer Counsel* policy since it implies the agent will ask all the other agents. However, the *Bounded Counsel* is more robust; even though an agent with less cases tends to ask more often counsel we see in Figure 3 that the computation time is decreasing. The fact is that the first component (retrieval time) keeps decreasing with smaller case bases and the increasing number of counsels that are required are sufficiently restricted by the *Bounded Counsel* policy to prevent a worsening of the cost in computation time.

We have also tested the system with a subset of the whole case base consisting in 120 sponges pertaining to the same three biological orders. Given the reduced number of cases, in these experiments we have only considered a 3 agents system. As we can see in table 2 the results are similar to those obtained using 280 cases, including the cost reduction of the *Bounded Counsel* and *Peer Counsel* compared to Committee. The gain in efficiency is also observable in this case, and we can compare the time spent per problem using the Committee policy (1.88 sec) with the lower one needed by the *Bounded Counsel* policy (0.93 sec), that is very close to the time need in the isolated agents scenario.

Finally, as an extreme scenario of agents with very few cases we tested the Committee policy (being the most robust) with 16 agents. In this scenario, with 280 sponges using 10-fold cross-validation, an agent's case base has only about 17 sponges—i.e. about 5.6 examples that randomly belong to 3 solution classes. In this scenario the Committee policy still has an accuracy of about 83% while the isolated agents are only capable of about 64% accuracy. If we move towards a scenario of 25 agents (with about 11 cases per case base) the Committee policy still has an accuracy of about 82% while the isolated agents are only capable of about 60% accuracy. Clearly, in these extreme scenarios the other policies

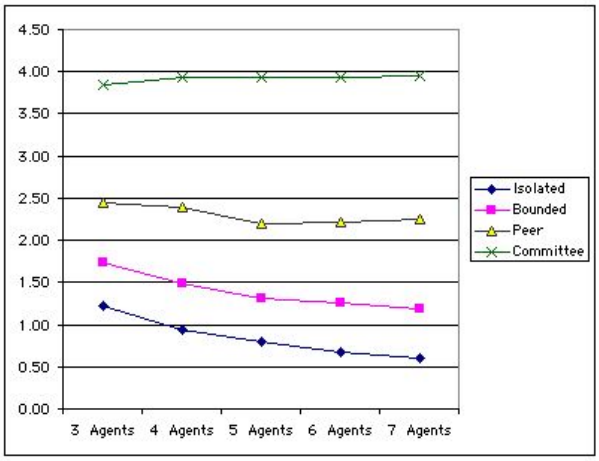

Fig. 3. Average time needed (in seconds) to solve a problem in the reported experiments.

(*Bounded Counsel* and *Peer Counsel*) are not appropriate. In fact, they can only make sense if the agents always assess their lack of self-competence and ask the other agents (that also will have very low self-competence assessments)—i.e. if both policies tend to work as the Committee policy asking counsel for every case to every agent.

5 Related Work

A general result on multiple model learning [4] demonstrated that if uncorrelated classifiers with error rate lower than 0.5 are combined then the resulting error rate must be lower than the one made by the individual classifiers. The BEM (*Basic Ensemble Method*) is presented in [7] as a basic way to combine continuous estimators, and since then many other methods have been proposed: *Stacking generalization*, *Cascade generalization*, *Bagging* or *Boosting* are some examples. However, all these methods do not deal with the issue of "partitioned examples" among different classifiers as we do—they rely on aggregating results

Policy	μ	σ	*Time*	*Cost*
Isolated Agents	82.00	12.84	0.64 sec	1.00
Bounded Counsel	87.99	10.29	0.93 sec	1.48
Peer Counsel	88.83	9.40	1.08 sec	1.71
Committee	88.99	8.40	1.88 sec	3.00

Table 2. Average precision and standard deviation for 3 agents and a case base of 120 sponges pertaining to three classes. All the results are obtained using a 10-fold cross validation.

from multiple classifiers that have access to *all* data. Their goal is to use multiplicity of classifiers to increase accuracy of existing classification methods. Our goal is to combine the decisions of autonomous classifiers (each one corresponding to one agent), and to see how can they cooperate to achieve a better behavior than when they work alone.

Usually, lazy learners (e.g. Nearest Neighbour) don't get benefited by any of the combination methods like *Bagging* or *Boosting*. Ricci and Aha propose in [10] a method to combine the decision of lazy learners. They create various NN classifiers, each one considering a different subset of features and then combine their results using ECOCs(Error-Correcting Output Codes). However, the case-base is not partitioned and each classifier works with all instances in the case-base.

When trying to combine decision through a voting scheme, the simplest way is a non-weighted voting combination, where all the agents have the same strength in the decision. The Naive Bayesian Classifiers (NBC) committees [12] proposal trains various NBC with the same training set but using different subsets of attributes, and then combine their predictions using some strategy (majority, etc). Again this approach does not deal with "partitioned examples". Moreover, usually the agent members of a committee can only cast a vote for one of the solution classes. The *Committee* CBR Policy is innovative in that the agents following it assign a fractional vote to each solution class in function of the number of relevant precedent cases found endorsing them.

The meta-learning approach in [3] is applied to partitioned data. They experiment with a collection of classifiers which have only a subset of the whole case base and they learn new meta-classifiers whose training data are based on predictions of the collection of (base) classifiers. They compare their meta-learning approach results with weighted voting techniques. The final result is an *arbitrator tree*, a centralized and complex method whose goal is to improve classification accuracy. We also work on "partitioned examples" but we assume no central method that aggregates results; moreover we assume a multiagent approach where communication and cooperation may have a cost that has to be taken into account.

Smyth and McKenna [11] propose a method to assess the competence of a case-base based on case coverage, case-base size and distribution of the cases. They measure the competence by finding *Competence Groups* (clusters of related cases) and measuring how each *Competence Group* is covered in the case-base. In our experiments, each agent compute its *self-competence* as a function of the number of cases endorsing the possible solution classes returned. That is because agents only need to assess the confidence degree of the answer to a specific problem. In [6] we propose an inductive technique that allows each agent to learn autonomously an individual *self-conpetence* function.

6 Conclusions and Future Work

We have presented a framework for cooperative CBR in multiagent systems. The framework is cooperative in that the CBR agents help each other to improve their individual performance. Since the agents improve with respect to their performance as isolated individual, cooperating is also in their individual interest—specially since the framework allows them to keep confidential their own cases. A major theme in multiagent systems is the *autonomy* of the agents. In our framework the agent autonomy is mainly ensured by two facts: i) the capability of each agent to determine whether or not itself is competent to solve a problem, and ii) the capability of each agent to integrate into a global solution for a problem the counsels given by other agents. In the experiments we have presented all agents used the same methods to implement these two capabilities. However, this option is just an experiment design decision, and a particular application that requires different biases by different agents is compatible with our framework.

Another issue is the generality of the cooperation policies and their dependence upon the CBR agents using LID. The cooperation policies depend only on the CBR agents being able to provide SERs (Solution Endorsement Records), so any CBR method that can provide that is compatible. For instance, CBR agent using k-nearest neighbour as a retrieval method could provide a SER for the k closest cases.

A natural evolution of this experimental setting is moving towards case-bases with higher volume in which learning *competence models* [9] of the involved agents is necessary. In higher volume case-bases the need for a distributed multiagent approach seems more practical than a completely centralized schema. A major difference from our current experimental setting is that the hypothesis that all agents have an unbiased sample of the data no longer holds. That is to say, with higher volume case-bases a CBR agent may be biased towards solving accurately a subset of all possible problems. For instance, a particular agent may solve often sponge identification problems *inside* the order of Astroforida—and very seldom problems inside the order of Axinellida. In this setting future work will investigate how CBR agents can learn *competence models* of other agents—e.g. which agent is competent in identifying a sponge inside the order of Astroforida. Competence models can guide an agent into asking counsel only (or mainly) to competent agents and to have this information into account when integrating solutions proposed by several agents. A relevant question here is whether it is possible to adapt the method in [11] to obtain the competence models of the other agents keeping the autonomy and the privacy of the individual case-bases.

Finally, we plan to lift the restriction of the case bases of the agents in a $\mathcal{M}$AC system being disjunct. Basically, our idea is that agents could incorporate in their case bases some cases originally owned by other agents. The interesting question here is this: what strategy of case sharing can improve the overall $\mathcal{M}$AC system performance —without every agent having in their case base every case known to the $\mathcal{M}$AC system.

Acknowledgements

The authors thank Josep-Lluís Arcos and Eva Armengol of the IIIA-CSIC for their support and for the development of the Noos agent platform and the LID CBR method respectively. Support for this work came from CIRIT FI/FAP 2001 grant and projects TIC2000-1414 "eInstitutor" and IST-1999-19005 "IBROW".

References

[1] Josep Lluís Arcos and Ramon López de Mántaras. Perspectives: a declarative bias mechanism for case retrieval. In David Leake and Enric Plaza, editors, *Case-Based Reasoning. Research and Development*, number 1266 in Lecture Notes in Artificial Intelligence, pages 279–290. Springer-Verlag, 1997.

[2] E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In *Submitted*, 2001.

[3] Philip K. Chan and Salvatore J. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *Proc. 12th International Conference on Machine Learning*, pages 90–98. Morgan Kaufmann, 1995.

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

[5] Ramon López de Mántaras. A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 6:81–92, 1991.

[6] S. Ontañón and E. Plaza. Learning when to collaborate among learning agents. In *Submitted*, 2001.

[7] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hydrid neural networks. In *Artificial Neural Networks for Speech and Vision*. Chapman-Hall, 1993.

[8] E. Plaza, J. L. Arcos, P. Noriega, and C. Sierra. Competing agents in agent-mediated institutions. *Journal of Personal Technologies*, 2:212–220, 1998.

[9] Enric Plaza, Josep Lluís Arcos, and Francisco Martín. Cooperative case-based reasoning. In Gerhard Weiss, editor, *Distributed Artificial Intelligence Meets Machine Learning. Learning in Multi-Agent Environments*, number 1221 in Lecture Notes in Artificial Intelligence, pages 180–201. Springer-Verlag, 1997.

[10] Francesco Ricci and David W. Aha. Error-correcting output codes for local learners. In *European Conference on Machine Learning*, pages 280–291, 1998.

[11] Barry Smyth and Elizabeth McKenna. Modelling the competence of case-bases. In *EWCBR*, pages 208–220, 1998.

[12] Z. Zheng. Naive bayesian classifier committees. In *Proceedings of the 10th European Conference on Machine Learning, ECML'98*, volume 1398 of *LNAI*, pages 196–207. Springer Verlag, 1998.