A bartering approach to improve multiagent learning

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

Multiagent systems offer a new paradigm to organize AI Applications. We focus on the application of Case-Based Reasoning to Multiagent systems. CBR offers the individual agents the capability of autonomously learn from experience. In this paper we present a framework for collaboration among agents that use CBR. We present explicit strategies for case bartering that address the issue of agents having a biased view of the data. The outcome of bartering is an improvement of individual agent performance and of overall multiagent system performance that equals the ideal situation where all agents have an unbiased view of the data. We also present empirical results illustrating the robustness of the case bartering process for several configurations of the multiagent system and for three different CBR techniques.

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. CBR offers the multiagent system paradigm the capability of autonomously learn from experience. In this paper we present a framework for collaboration among agents that use CBR and some experiments illustrating how they can improve its performance using case bartering.

The individual case bases of the CBR agents are the main issue here, if they are not properly maintained, the overall system behavior will be suboptimal. In a real system, there will be agents that can very easily obtain certain kind of cases, and that will very costly obtain other types of cases, and for sure that other agents in the system will be in the inverse situation. It will be beneficial for both agents if they reach an agreement to trade cases. This is a very well known strategy in the human history called *bartering*. Using case bartering, agents that have a lot of cases of some kind will give them to another agents in return to more interesting cases for them.

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. Separate case bases make sense for different reasons like privacy or efficiency. If the case bases are owned by some organizations, perhaps they are not willing to donate the contents of its case bases to a centralized one where CBR can be applied. Moreover, in the case that the case bases where not private, more problems can arise from having all the cases in a single one, such as efficiency or storage problems.

In this paper we focus on case bartering. We present two protocols for case bartering that improve the overall performance of the system and of the individual CBR agents without compromising the agent's autonomy and keeping individual and private case bases. This protocols will try to minimize the individual case base bias (how far is a case base of being a good sample of the overall distribution of cases).

The structure of the paper is as follows. Section 2 presents the collaboration scheme that the agents use, then the individual case base bias measurement is introduced in section 3. After that, section 4 presents the case bartering mechanism, including the bartering protocols. Finally, The experiments are explained in section 5 and the paper closes with related work and conclusion sections.

2. COLLABORATION SCHEME

A multiagent CBR ($\mathcal{M}$AC) system $\mathcal{M} = \{(A_i, C_i)\}_{i=1...n}$ is composed on *n* agents, where each agent A_i has a case base C_i. In the experiments reported here we assume that initially case bases are disjunct ($\forall A_i, A_j \in \mathcal{M} : C_i \cap C_j = \emptyset$), i.e. initially there is no case shared by two agent's case bases. 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 $K = \{S_1 \dots S_K\}$. 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 CBR method of 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.

2.1 Voting Scheme

The voting scheme defines the mechanism by which an agent reaches an aggregate solution from a collection of SERs coming

AAMAS'02, July 15-19, 2002, Bologna, Italy.

$\mathcal{M}$AC ICB	*3 Ag.*	*5 Ag.*	*8 Ag.*	*10 Ag.*
0.0	88.36%	88.12%	87.50%	86.75%
0.1	86.07%	87.50%	85.35%	85.00%
0.2	81.46%	83.53%	83.00%	82.00%

Table 1: Classification accuracy for the marine sponge classification problem for systems with several mean *ICB* bias.

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 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

$$Vote(S_k, A_j) = \frac{E_k^j}{c+N} = \frac{N}{c+N}\frac{E_k^j}{N}$$

where $N = \sum_{r=1\ldots K} E_r^j$ is the total number of cases retrieved and 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. In the rightmost formulation we can distinguish two factors, the first one ($N/(c+N)$) favors the agents with a high number of retrieved cases, and the second one (E_k^j/N) fractionally assigns the vote to the different classes.

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 the class with more votes in total.

This voting scheme can be seen as a variation of *Approval Voting* [2]. In *Approval Voting* each agent vote for all the candidates they consider as posible solutions without giving any weight to its votes. In our scheme, *Approval Voting* can be implemented making $Vote(S_k, A_j) = 1$ if $E_k^j \neq 0$ and 0 otherwise.

There are two differences between the standard *Approval Voting* and our voting scheme. The first one is that in our voting scheme agents can give a weight to each one of its votes. The second difference is that the sum of the votes of an agent is bounded to 1. Thus we can call it *Bounded-Weighted Approval Voting* (BWAV).

We will show now the $Committee$ collaboration policy that uses this voting scheme (see [8] for a detailed explanation and comparison of several collaboration policies).

2.2 Committee Policy

In this collaboration policy the agent members of a $\mathcal{M}$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 the 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.

3. CASE BASE BIAS

In a previous work [8] we have shown how agents can obtain better results using the *Committee* collaboration policy that working alone. However, in those experiments we assumed that every agent had a representative sample of cases in its individual case base. When an agent has a case base that is not representative of the overall distribution, we say that the agent has a *biased* case base.

In this section we are going to define a measure of the degree of biasing of an individual case base (*ICB* bias or *Individual Case Base bias*), then we will show how the performance of the *Committee* degrades as the *ICB* bias of the agents grow. Later sections introduce bartering policies to improve the *Committee* performance.

3.1 Individual Case Base Bias

Let be $d_i = \{d_i^1, \ldots, d_i^K\}$ the individual distribution of cases for an agent A_i, where d_i^j is the number of cases with solution $S_j \in K$ in the the case base of A_i. Now, we can estimate the overall distribution of cases $D = \{D^1, \ldots, D^K\}$ where D^i is the estimated probability of the class S_i, $D^j = \sum_{i=1}^n d_i^j / \sum_{i=1}^n \sum_{l=1}^K d_i^l$.

To measure how far is the case base C_i of a given agent A_i of being a representative sample of the overall distribution we will define the *Individual Case Base* (*ICB*) bias, as the square distance between the distribution of cases *D* and the (normalized) individual distribution of cases obtained from d_i:

$$ICB(C_i) = \sum_{l=1}^{K} \left(D^l - \frac{d_i^l}{\sum_{j=1}^K d_i^j} \right)^2$$

In order to see how the *ICB* bias affects the performance of the system, Table 1 shows the accuracy of several multiagent systems with increasing *ICB* bias (the $\mathcal{M}$AC *ICB* bias represents the average bias in the system and is calculated as the mean of all the *ICB* bias of the individual agents in the system). There we can see that when the agents have case bases that are not representative (those with a high *ICB*) the agents using the *Committee* policy obtain lower accuracies. In the following sections, we will show how case bartering improves accuracy by lowering the individual biases.

4. CASE BARTERING

In the physical world, bartering involves the interchange of two goods. But as our agents will barter with cases (that are just information) they will have the option of send only a copy of the cases to the other agents without losing them. We have experienced with two different strategies of bartering: in the first one the agents send only copies of the cases (*copy mode*), and in the second one, the agents forget the cases they send (*non-copy mode*). The main difference between these two cases is wether the sender retains the cases or not. Experiments showing the results of applying both strategies are explained in section 5.

In this section, we are going to present the Case Bartering protocol that the agents use in order to improve the overall performance. These protocols are independent of the strategy used by the agents, they only manage how the agents can reach bartering agreements. But first, we are going to explain how two agents can reach a bartering agreement and the concrete strategies that the agents are using in our experiments.

4.1 Case Bartering Mechanism

To reach an agreement for bartering between two agents, there must be an offering agent A_i that sends an offer to another agent A_j. Then A_j has to evaluate whether the offer of interchanging cases with A_i is interesting, and accept or reject the offer. If the offer is accepted, we say that A_i and A_j have reached a bartering agreement, and they will interchange the cases in the offer.

Formally an offer is a tuple $o = \langle A_i, A_j, S_{k_1}, S_{k_2}\rangle$ where A_i is the offering agent, A_j is the receiver of the offer, and S_{k_1} and S_{k_2}

are two solution classes, meaning that the agent A_i will send one of its cases (or a copy of it) with solution S_{k_2} and A_j will send one of its cases (or a copy of it) with solution S_{k_1}.

4.2 Making and accepting offers

The Case Bartering Protocol is not restrictive in how many offers can an agent send at a given time. So, many strategies can be used here, but in our experiments, the agents use a very simple one to choose which are the most interesting offers, as follows for a given agent A_i:

- For each solution class $S_{k_1} \in \{S_1 \dots S_K\}$
- A_i looks if increasing by one its number of cases with solution S_{k_1} will decrease its *ICB* bias.
- If so, A_i chooses which agent A_j of the others is the best one to ask for cases of solution S_{k_1} (Currently the chosen A_j is the one with more cases of the solution class S_{k_1}).
- In this step we have to distinguish two modes
  1. In the *copy mode*, A_i determines which is its best class S_{k_2} (currently the chosen one is the class for which it has more cases), and makes the offer $o = \langle A_i, A_j, S_{k_1}, S_{k_2} \rangle$, i.e. A_i offers to A_j a case of solution S_{k_2} if A_j gives one of solution S_{k_1} to A_i.
  2. In the *non-copy mode*, A_i determines, from the subset of classes that decreasing by one its number of cases of that class will decrease the *ICB* bias, which is its best class S_{k_2} (currently the chosen one is the class for which it has more cases), and makes the offer $o = \langle A_i, A_j, S_{k_1}, S_{k_2} \rangle$, i.e. A_i offers to A_j a case of solution S_{k_2} if A_j gives one of solution S_{k_1} to A_i.

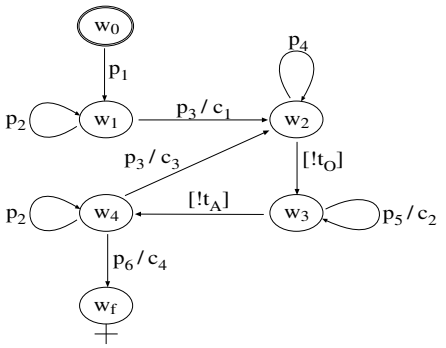
When an agent receives a set of offers, it also has to choose which of these offers to accept and which not. In our experiments the agents use the simple rule of accepting every offer that reduces its own *ICB* bias. Thus, we will define the set of interesting offers *Interesting*(O, A_i) of a set of offers O for an agent A_i as those offers that will reduce the *ICB* bias of A_i. Moreover, in the *copy mode* an agent should not send twice the same case to the same agent. So, the agents will only accept those interesting offers that can satisfy this constraint (i.e. can provide a new case).

4.3 Case Bartering Protocol

We are going to present two different protocols for Case Bartering, both synchronous (there are preestablished stages ("rounds") where the agents can send their offers, then the protocol moves to the next stage, etc). The first one is called the *Simultaneous Case Bartering Protocol*, and the second one the *Token-Passing Case Bartering Protocol*.

When an agent member of the $\mathcal{MAC}$ wants to enter in the bartering process, it sends an initiating message to all the other agents in the $\mathcal{MAC}$. Then all the other agents answer whether or not they enter the bargaining process. This initiating message contains the parameters for bartering: a parameter t_O, corresponding to the time period that the agents have to make offers; a parameter t_A, corresponding to the time period that the agents have to send the accept messages; the number n of agents taking part in the bartering, and R_{max}, the maximum number of bartering rounds that the bartering will have. Once the agents have answered to this initial message, the bartering starts.

We have specified both protocols using the formalism used in ISLANDER [4] (also available online at *http://e-institutor.iiia.csic.es /e-institutor/publications.html*), and we also include a plain text explanation. The appendix briefly summarizes the notation of ISLANDER that we use in the following sections.



p_1:	Inform(?A_{ini},all,c_1=?$t_O \wedge c_2$=?$t_A \wedge c_3$=?$n \wedge c_4$=?R_{max})
p_2:	Inform(?A_i,all,?d_i)
p_3 / c_1:	Inform(!A_{ini},all,start) / | !w_1 w_1 d_i | = !n
p_4:	Offer(?A_i,?A_j,?o)
p_5 / c_2 :	Accept(?A_i?A_j,?o) / (?A_j,?A_i,?o) ∈ !w_2 w_2(A_i,A_j,o)
p_3 / c_3:	Inform(!A_{ini},all,start) / | !w_4 w_4 d_i | = !n ∧ | !w_3 w_3 o | ≠ 0 ∧ | !*w_4 w_4 A_i | < !R_{max}*!n
p_6 / c_4:	Inform(!A_{ini},all,bye) / | !w_4 w_4 d_i | = !n ∧ | !w_3 w_3 o | = 0 ∨ | !*w_4 w_4 A_i | ≥ !R_{max}*!n

Figure 1: *Simultaneous Case Bartering Protocol* specification

4.3.1 Simultaneous Case Bartering Protocol

In this protocol all the agents send their offers simultaneously in every round. After all the offers have been sent for the current round, all the agents send a message for the offers they accept. Figure 1 shows the formal specification of the protocol, and below we describe it in detail.

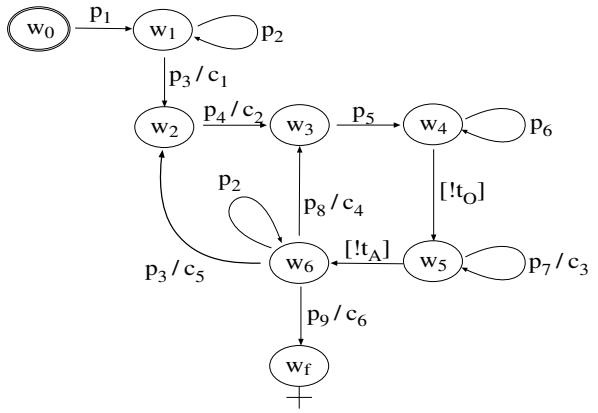
1. The initial state is w_0 (see Figure 1). First, the initiating agent (A_{ini}) sends a message (p_1) to all the other agents with the protocol parameters (t_O, t_A, n and R_{max}). then the protocol to move to state w_1.

2. Each agent broadcasts its individual distribution d_i (message p_2). The protocol remains at w_1 until each agent has done this, as established by condition c_1.

3. When all the agents have sent its individual distribution (condition c_1), they are able to compute the overall distribution estimation D. Now the initiating agent broadcasts a *start* signal in message p_3 and the protocol moves to state w_2.

4. In state w_2 the agents send their bartering offers using message p_4. When the time t_O is reached, the protocol moves to state w_3.

5. In state w_3 the agents send the accept messages (p_5) for those offers they are interested in. The condition c_2 establishes than an agent can only confirm those offers that where send to him in the state w_2. When the maximum time t_A is over, all the unaccepted offers are considered as rejected and the protocol moves to w_4.



p_1:	Inform(?A_{ini},all,c_1=?$t_O \wedge c_2$=?$t_A \wedge c_3$=?$n \wedge c_4$=?R_{max})
p_2:	Inform(?A_i,all,?d_i)
P_3 / c_1:	Inform(!A_{ini},all,new-round)
	$\mid$!w_1 w_1 d_i $\mid$ =!n
p_4 / c_2:	GiveToken(!A_{ini},?A_{token}) /
	?At_{token} = f(!w_1 w_1 (A_i d_i))
p_5:	Inform(!A_{token},all,start)
p_6:	Offer(!A_{token},?A_j,?o)
p_7 / c_3:	Accept(?A_i,!A_{token},?o) /
	(!A_{token},?A_i,?o) $\in$!w_4 w_4(A_{token},A_j,o)
p_8 / c_4:	GiveToken(!A_{token},?A_{token}) /
	$\mid$!w_6 w_6 d_i $\mid$ =!n $\wedge$ $\mid$!w_2 w_6 A_{token} $\mid$ < !n $\wedge$
	?A_{token} = f(!w_1 w_1 (A_i d_i) / ($A_i \notin$!w_2 w_6 A_{token}))
P_3 / c_5:	Inform(!A_{ini},all,new-round) /
	$\mid$!w_6 w_6 d_i $\mid$ =!n $\wedge$ $\mid$!w_2 w_6 A_{token} $\mid$ = !n $\wedge$
	$\mid$!w_2 w_5 o $\mid \neq 0 \wedge \mid$!*w_6 w_6 A_i $\mid$ < !R_{max}*!n
P_9 / c_6:	Inform(!A_{ini},all,bye) /
	$\mid$!w_6 w_6 d_i $\mid$ =!n $\wedge$ $\mid$!w_2 w_6 A_{token} $\mid$ = !n $\wedge$
	($\mid$!w_2 w_5 o $\mid$ = 0 $\vee \mid$!*w_6 w_6 A_i $\mid \geq$!R_{max}*!n)

Figure 2: *Token-Passing Case Bartering Protocol* specification

6. In state w_4 each agent broadcasts its new individual distribution d_i (message p_2). The protocol remains at w_4 until each agent has done this as established by conditions c_3 and c_4.

7. If there have been no interchanged cases or the maximum number of iterations R_{max} is reached (condition c_4), the Case Bartering Protocol ends (w_f), otherwise, when condition c_3 holds the agents start a new round and the protocol moves again to w_2 when the initiating agent broadcasts a *start* message. Notice, that the agents will know that there have been no interchanged cases when the individual distributions d_i do not change from one round to another.

4.3.2 Token-Passing Case Bartering Protocol

The main difference between this protocol and the previous one is the introduction of a Token-Passing mechanism, so that only the agent who has the Token can make offers to the others. Figure 2 shows the formal specification of the protocol, and below we describe it in detail.

1. The initial state is w_0. First, the initiating agent (A_{ini}) sends a message (p_1) to all the other agents with the protocol parameters (t_O, t_A, n and R_{max}). Then the protocol moves to state w_1.

2. Each agent broadcasts its individual distribution d_i (message p_2). The protocol remains at w_1 until each agent has done this, as established by condition c_1.

3. When all the agents have sent its individual distribution (condition c_1), they are able to compute the overall distribution estimation D. The initiating agent broadcasts a *new-round* signal in message p_3 and the protocol moves to state w_2.

4. Each agent is able to compute the *ICB* bias of all the agents taking part in the bartering (including itself), and this will define the order in which the token will be passed through. Now the initiating agent gives the token (message p_4) to the agent with the highest *ICB* bias (condition c_2) and the protocol moves to w_3. In the specification of condition c_2 we are assuming that we have a function f that given a list of pairs (A_i, d_i) (i.e. agent and *ICB* bias) returns the agent with higher *ICB* bias.

5. Now the agent A_{token} owning the token broadcasts a *start* signal in message p_5 and the protocol moves to w_4.

6. In state w_4 the agent A_{token} sends its bartering offers using message p_6. When the time t_O is reached, the protocol moves to state w_5.

7. In state w_5 the agents will send the accept messages (p_7) for those offers they are interested in. The condition c_3 establishes than an agent can only confirm those offers that where send to him in the state w_4. When the maximum time t_A is over, all the unaccepted offers are considered as rejected and the protocol moves to state w_6.

8. Each agent broadcasts its new individual distribution d_i (message p_2). The protocol remains at w_6 until each agent has done this, as established by conditions c_4, c_5 and c_6.

9. When all the agents have send its individual distributions, three different situations may arise:

   (a) If there are some agents that still haven't owned the Token in the current round (condition c_4), the owner of the token passes it to the next agent with message p_8 and the protocol moves to w_3 again. To specify condition c_4 we have used the same function f used in the specification of condition c_2.

   (b) If every agent has owned the token once in this round, there have been some interchanged cases and the maximum number of iterations R_{max} still has not been reached (condition c_5), the agents start a new round and the protocol moves again to w_2 when the initiating agent A_{ini} broadcasts a *new-round* signal in message (p_3).

   (c) If every agent has owned the token once in this round but there have been no interchanged cases or the maximum number of iterations R_{max} is reached (condition c_6), the Case Bartering Protocol ends (w_f) after the initiating agent A_{ini} sends a *bye* signal in message p_9.

4.4 Protocol discussion

To ensure the convergence of both protocols, we have to distinguish between the *copy mode* and the *non-copy mode*.

In the *copy mode*, we must have in mind the only restriction that we have imposed: an agent cannot send twice the same case to the same agent. With this restriction it is easy to see that both protocols cannot run indefinitely, because each agent has a limited number of cases to trade with. So, we can say that in a bounded number of rounds both protocols will end.

The above restriction cannot be used in *non-copy mode*, because the cases will never be replicated in the $\mathcal{MAC}$. In this situation, the

	Agent 1	*Agent 2*	*Agent 3*
Astrophorida	0.7	0.15	0.15
Hadromerida	0.15	0.7	0.15
Axinellida	0.15	0.15	0.7

Table 2: Probability of having cases of each solution class for the 3 agents scenario. This corresponds to an $\mathcal{M}$AC *ICB* of about 0.2

	3 Agents	*5 Agents*	*8 Agents*	*10 Agents*
Before	84.00	50.40	31.50	25.20
After SCBP	187.49	79.60	49.42	40.75
After TPCBP	183.00	77.50	46.78	38.15

Table 3: Comparison of the mean case base size before and after the bartering process in the *copy mode*.

safest way to ensure that the protocol will not run indefinitely is to preset the maximum number of rounds R_{max}.

Comparing the protocols, we can see that the *Simultaneous* protocol has the problem that an agent has to decide whether to accept offers or not without knowing if its own offers are going to be accepted. The *Token-Passing* protocol tries to solve this problem by allowing one agent only to send offers at a time.

Moreover, in the *Simultaneous* protocol, two agents may send the same offer to each other, i.e. A_i sends to A_j $\langle A_i, A_j, S_{k_1}, S_{k_2} \rangle$ and A_j sends to A_i $\langle A_j, A_i, S_{k_2}, S_{k_1} \rangle$. As they are identical, we want to treat them as if there was only one offer sent. We call this phenomena the *Identical Offer Clash* (*IOC*). When two agents detect an *IOC*, they behave considering that only one offer has been sent, and only one case of each class is bartered.

5. EXPERIMENTAL RESULTS

In this section we want to show how the classification accuracy of the CBR agents improve using the case bartering protocols with respect to systems where the CBR agents do not use them. We also show results concerning case base sizes after the bartering and the number of rounds needed to converge to a stable case distribution.

We use the marine sponge classification problem as our test bed. 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, benthology specialists are distributed around the world and they have experience in different benthos that spawn species with different characteristics due to the local habitat conditions.

In order to show the improvements obtained in the system when the agents use case bartering, 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*). In an experimental run, cases are randomly distributed among the agents (e.g. if the training set is composed of 252 cases and we have a 4 agents system, each agent will receive about 63 cases). In the testing phase, problems arrive randomly to one agent in the $\mathcal{M}$AC. The goal of the agent receiving a problem is to identify the correct biological order given the description of a new sponge. Once an agent has received a problem, the $\mathcal{M}$AC will use the *Committee* collaboration policy to obtain the prediction.

For experimentation purposes, we force biased case bases in every agent by increasing the probability of each agent to have cases of some classes and decreasing the probability to have cases of some other classes. For example, Table 2 shows the probabilities for 3 agents of having cases for each solution class.

In order to test the generality of the protocols, we have tested them using systems with 3, 5, 8 and up to 10 agents, and using several CBR methods: nearest neighbor, 3-nearest neighbor and LID[1]. The results presented here are the average of 5 10-fold cross validation runs.

The figures 3, 4 and 5 show the results of applying the two case bartering protocols. The left part of those figures show the average accuracy obtained for the *non-copy mode*, and the right part show the average accuracy obtained for the *copy mode*. Three bars are shown for each scenario, the *biased* results represent the average accuracy obtained by the $\mathcal{M}$AC without using case-bartering with biased individual case bases; and the *SCBP* and *TPCBP* results represent the average accuracy obtained by the $\mathcal{M}$AC after using the *Synchronous Case Bartering Protocol* and *Token-Passing Case Bartering Protocol* respectively. We can see in those figures that in all the scenarios, the $\mathcal{M}$AC systems using case bartering obtain a significative gain in accuracy than those systems that do not use case bartering. This shows the independence of the bartering protocols from the CBR method used by the individual agents. Those figures also show that case bartering is robust even when the size of the case bases decreases and the number of cases an agent can barter is very small, as we can see for the 10 agents scenario where each agent has only about 25 cases (i.e. less than 9 cases per class).

Comparing the accuracy obtained by the two protocols *SCBP* and *TPCBP* we see that both have nearly the same accuracy in all the scenarios. We can see that there is never a difference greater than 1% between the results of the Simultaneous protocol and the results of the Token-Passing protocol. Therefore no bartering protocol is significantly better than another but both are significantly better than using no bartering protocol.

Figure 3 shows that when the agents use simple methods like nearest neighbor, they can take more advantage of the bartering process than when the agents use more sophisticated methods like LID (Figure 5). For instance, we can see how in a 3 agent scenario (with about 84 cases per agent) using nearest neighbor, the accuracy can boost from 81.43% to 90.71% in the *copy mode* or to 89.21% in the *non-copy mode*. In the same 3 agent scenario but using LID the accuracy for the system without using case bartering is 84.82%, and after case bartering it raises to 90.36% in the *copy-mode* or 89.36% in the *non-copy mode*. The gain in accuracy for the nearest neighbor is higher than for LID, but it is due to the fact that the accuracy obtained by the system without using case-bartering is lower for nearest neighbor than for LID, and the accuracies obtained after case bartering are more similar. Therefore, we can conclude that LID is more resilient to biased case bases and that bartering is positive to all three CBR methods but specially useful for nearest-neighbor methods.

Looking at Figure 4, we can see that 3-nearest neighbor results are worse than with nearest neighbor or LID specially for the *non-copy mode* and even more when the individual case base sizes are small (in the 8 or 10 agent scenario). The reason is that 3-nearest neighbor do not work well when the case bases are small. Thus, we can see that in the *non-copy mode* the results of 3-nearest neighbor are not very good because the case base sizes keep being small, but when we allow case redundancy (in the *copy mode*) and the case bases are bigger, 3-nearest neighbor starts behaving better and obtains nearly the same results as nearest neighbor.

Table 3 shows the case base sizes reached after case bartering in the *copy mode*. We see that the agents stop interchanging cases before each agent acquires all known cases in the system. Moreover, except in the 3 agents scenario, the case base sizes do not increase

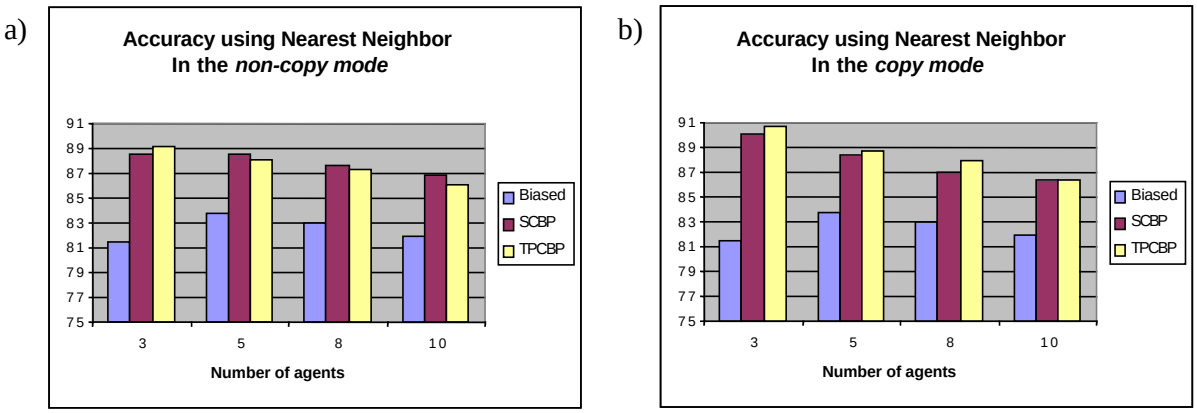


Figure 3: Accuracy plot for systems where the agents use nearest neighbor

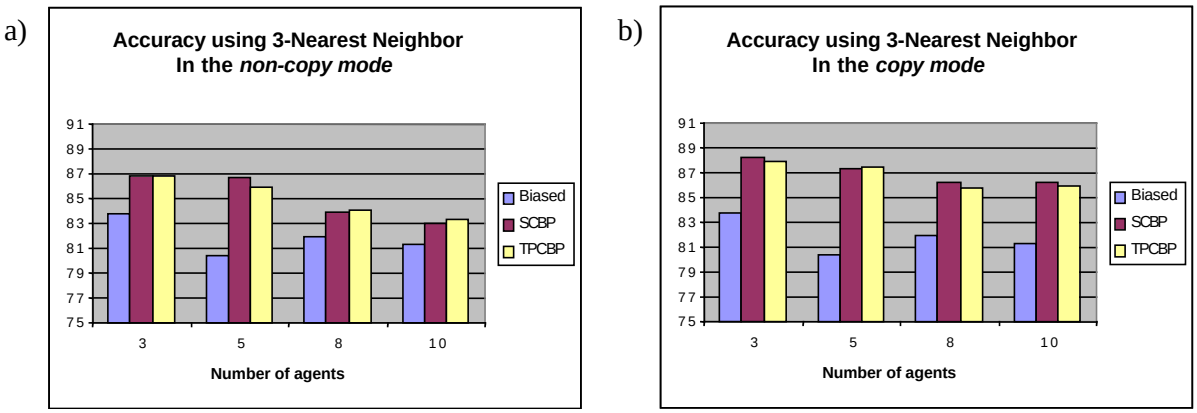


Figure 4: Accuracy plot for systems where the agents use 3-nearest neighbor

very much. The 3 agents scenario is special because the initial case bases of the agents are quite big, and to repair their *ICB* biases the number of cases needed to be bartered is quite higher than in the 5, 8 or 10 agent scenarios. We also see that the case base sizes in the *non-copy mode* obtained using the Token-Passing protocol are slightly smaller than the ones obtained using the Simultaneous protocol.

Concerning the convergence of the protocols, they always converge, and the maximum number of rounds R_{max} (set to 300 in these experiments) is practically never reached. Table 4 shows the average number of iterations of the protocols needed to converge to a stable case distribution for the *copy mode*. We can see there that the Simultaneous protocol is much faster than the Token-Passing one (taking only in consideration the number of rounds needed to converge). The reason is that in the Token-Passing protocol only one agent can make offers each round, so it needs about n times more rounds (being n the number of agents in the system) than the Simultaneous protocol. For comparison purposes the third row of Table 4 shows the number of rounds needed by the Simultaneous protocol multiplied by n. Table 5 shows the average number of iterations of the protocols needed to converge for the *non-copy mode*. Again the Simultaneous protocol needs less rounds to converge than the Token-Passing protocol for the same reason. We can see comparing both tables (4 and 5) that the number of rounds needed to converge in the *non-copy mode* is smaller than in the *copy mode*. This is as expected, since in *non-copy mode* the bias of individual case bases is modified in two ways: by the case an agent loses by sending it away and the new case that arrives — while copy mode only has the latter effect.

	3 Agents	*5 Agents*	*8 Agents*	*10 Agents*
SCBP	79.4	24.9	14.0	12.9
TPCBP	212.0	101.1	97.2	99.4
*SCBP*n*	238.2	124.5	112	129

Table 4: Number of rounds need to converge in the case bartering protocols for the *copy mode*.

	3 Agents	*5 Agents*	*8 Agents*	*10 Agents*
SCBP	19.0	10.7	9.5	5.6
TPCBP	36.0	40.0	61.3	50.2
*SCBP*n*	57	53.5	76	56

Table 5: Average number of rounds need to converge in the case bartering protocols for the *non-copy mode*.

Comparing now the two modes, we see that in the *non-copy mode* the $\mathcal{MAC}$ obtains lower accuracies than in the *copy mode*. But, on the other hand, in the *non-copy mode*, the average number of cases per agent does not increase and in the *copy mode* the size of the individual case bases grows. Therefore, we can say that in the *copy mode* (when the agents send copies of the cases without forgetting them) the agents obtain greater accuracies, but at the cost of increasing the individual case base sizes. In other words, they

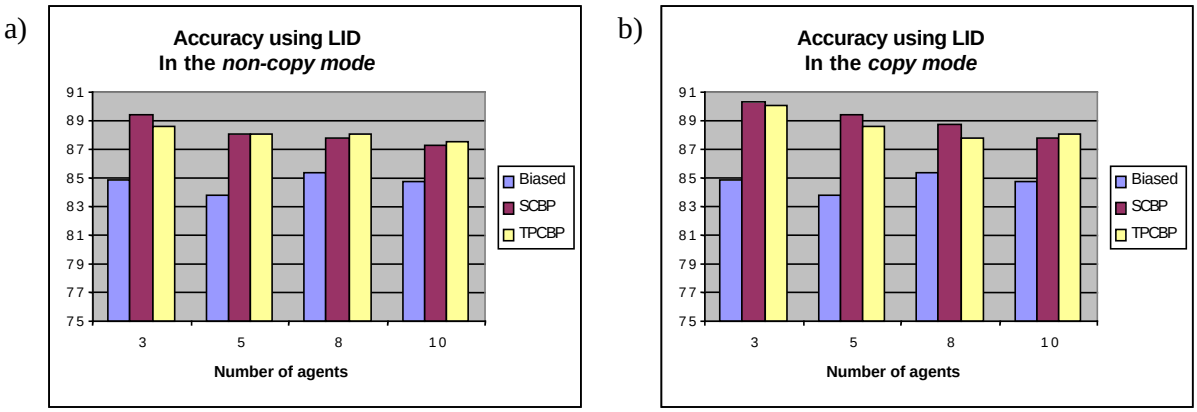


Figure 5: Accuracy plot for systems where the agents use LID

improve the accuracy allowing case redundancy in the contents of individual case bases (a case may be contained in more than one individual case base), while in the *non-copy mode* the agents only reallocate the cases but allowing only a single copy of each case in the system.

Finally, Table 6 shows the average accuracy for a individual agent before and after the case bartering process in *copy mode*. We can see that the barternig process also improves the individual performance of agents with respect to the biased situation; therefore, bartering makes sense for individual agents whether or not they are willing to cooperate. However, since using the *Committe* collaboration policy the performance is clearly better it makes sense for individual to engage in such a collaboration.

6. RELATED WORK

Several areas are related to our work: multiple model learning (where the final solution for a problem is obtained through the aggregation of solutions of individual predictors), case base competence assessment, and negotiation protocols. Here we will briefly describe some relevant work in these areas that is close to us.

A general result on multiple model learning [7] 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 [9] as a basic way to combine continuous estimators, and since then many other methods have been proposed: *Cascade generalization* [6], *Bagging* [3] or *Boosting* [5] 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 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 intention 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. A more similar approach is the one proposed in [13], where a MAS is proposed for pattern recognition. Each agent is a specialist recognizing only a subset of all the patterns, and the predictions are combined dynamically.

Learning from biased data sets is a well known problem, and many solutions have been proposed. Vucetic and Obradovic [12] propose a method based on a bootstrap algorithm to estimate class probabilities in order to improve the classification accuracy. However, their method does not fit our needs, because they need the entire test set available for the agents before start solving any problem in order to make the class probabilities estimation.

Related work is that of case base competence assessment. We use a very simple measure comparing individual with global distribution of cases; we do not try to assess the areas of competence of (individual) case bases - as proposed by Smyth and McKenna [11]. This work focuses on finding groups of cases that are competent.

In [10] Schwartz and Kraus discuss negotiation protocols for data allocation. They propose two protocols, the sequential protocol, and the simultaneous protocol. These two protocols can be compared respectively to our *Token- Passing Case Bartering Protocol* and *Simultaneous Case Bartering Protocol*, because in their simultaneous protocol, the agents have to make offers for allocating some data item without knowing the other's offers, and in the sequential protocol, the agents make offers in order, and each one knows which were the offers of the previous ones.

7. CONCLUSIONS AND FUTURE WORK

We have presented a framework for cooperative Case-Based Reasoning in multiagent systems, where agents use a market mechanism (bartering) to improve the performance both of individuals and of the whole multiagent system. The agent autonomy is maintained, because if an agent do not want to take part in the bartering, he just has to do nothing, and when the other agents notice that there is one agent not following the protocol they will ignore it during the remaining iterations of the bartering process.

In this article we have shown a problem arising when data is distributed over a collection of agents, namely that each agent may have a skewed view of the world (the individual bias). Comparing empirical results in classification tasks we saw that both the individual and the overall performance decreases when bias increases. The process of bartering shows that the problems derived from distributed data over a collection of agents can be solved using a market-oriented approach. Each agent engages in a barter only when it makes sense for its individual purposes but the outcome is an improvement of the individual and overall performance.

The naive way to solve the ICB bias problem could be to centralize all data in one location or adopt a completely cooperative multiagent approach where each agent sends its cases to other agents and they retain what they want (a "gift economy"). The problem with the completely cooperative approach is that the outcome improves but redundancy also increases and there may be scaling up problems; the bartering approach tries to interchange cases only to the amount that is necessary and not more.

	NN			3-NN			LID		
	Biased	*SCBP*	*TPCBP*	*Biased*	*SCBP*	*TPCBP*	*Biased*	*SCBP*	*TPCBP*
3 Agents	73.35	89.25	88.21	59.52	76.50	76.60	73.80	82.14	87.14
5 Agents	67.10	82.60	82.50	56.25	61.92	63.57	68.38	79.52	80.35
8 Agents	65.40	78.87	76.53	48.80	56.78	57.14	67.50	75.00	76.07
10 Agents	65.15	78.57	73.21	46.78	54.28	53.75	66.66	69.64	73.00

Table 6: Individual accuracy of agents before (*Biased*) and after the case bartering (*SCBP* and *TPCBP*) processes.

We have presented the copy and non-copy modes for modelling two different but related scenarios in multiagent case-based reasoning. In the *non-copy mode* cases are viewed as *assets* and ownership of cases requires that the acquisition of a case by an agents requires the other agent to relinquish any possession or use rights on that case. This mode insures that redundancy of cases in the multiagent system will not increase. However, a limited amount of redundancy can improve the multiagent system performance so if circumstances permit it is a good idea to allow it. This is precisely what the *copy mode* allows; however, *copy mode* is applicable only in scenarios where cases are seen as *information* (and not as assets) that can be shared when it is in the interest of the sharing partners.

In the experiments reported in this paper, the agents use strategies for choosing which offers to generate and send to other agents and for choosing which offers to accept from other agents. Currently, both strategies try to minimize the *ICB* bias measure. The *ICB* bias estimates the difference between the individual and global case distribution over the classes. However, we plan to study other kinds of biases that may characterize the individual agents' case base. In order to compute these new bias measures, the agents may need to make public more information. Thus, a modification in the bartering protocols would be needed to manage the information required. Moreover, the *ICB* bias measure presented in this paper is only useful when we are dealing with classification tasks. Therefore, new bias measures are needed to manage non-classification tasks.

We have focused on bartering for agents using lazy learning but future work should address the usefulness of bartering for eager (inductive) learning techniques.

Acknowledgements

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

8. REFERENCES

[1] E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In *12th European Conference on Machine Learning*, 2001.

[2] S. J. Brams and P. C. Fishburn. *Approval Voting*. Birkhauser, Boston, 1983.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] M. Esteva, J. Padget, and C. Sierra. Formalising a language for institutions and norms. In *Intelligent Agents VIII, Proceedings ATAL'01*, LNAI. Springer Verlag, To appear.

[5] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th ICML*, pages 148–146. Morgan Kaufmann, 1996.

[6] J. Gama. Local cascade generalization. In *Proc. 15th International Conf. on Machine Learning*, pages 206–214. Morgan Kaufmann, San Francisco, CA, 1998.

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

[8] S. Ontañón and E. Plaza. Learning when to collaborate among learning agents. In *12th European Conference on Machine Learning*, 2001.

[9] 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.

[10] R. Schwartz and S. Kraus. Bidding mechanisms for data allocation in multi-agent environments. In *Agent Theories, Architectures, and Languages*, pages 61–75, 1997.

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

[12] S. Vucetic and Z. Obradovic. Classification on data with biased class distribution. In *12th European Conference on Machine Learning*, 2001.

[13] L. Vuurpijl and L. Schomaker. A framework for using multiple classifiers in a multiple-agent architecture. In *Third International Workshop on Handwriting Analysis and Recognition*, 1998.

APPENDIX

A. PROTOCOL SPECIFICATION

Four basic elements shape the protocol specification formalism: *States*, *Transitions* (connecting two states in an oriented way), *Performatives* (containing a sender, the receivers, and optionally some content parameters), and *Conditions* (specifying whether a performative is valid or not depending on some restrictions). In order to specify all the above, we have used the following notation:

A state is noted w_i. A *Performative* is noted p_i, and is used with the form: *Performative(sender, receiver, ...)*. A *Condition* is denoted as c_i, and is composed of a set of constraints linked by conjunctive or disjunctive relations.

Each *Transition* is labeled by a *Performative* and optionally a *Condition* (p_i/c_j) meaning that the *Performative* p_i is only valid when the *Condition* c_j is satisfied. Some *Transitions* are labeled with a *Timeout* ($[!t]$) instead of a *Performative*, meaning that when the time t is reached after the arrival to the current state, the transition is in effect and the system moves to the next state.

A variable is noted $?x$. When we want to denote the value of a previously instantiated variable, we write $!x$. And when we want to obtain the set of values that a variable has taken we write: $!w_i w_j x$; this expression denotes the set of values taken by the variable x during the last transition of the protocol from w_i to w_j. And if we want to obtain the set of all the instantiations of a variable from the beginning of the protocol, we write $^*!x$.