Learning and Joint Deliberation through Argumentation in Multi-Agent Systems

Santi Ontañón
CCL, Cognitive Computing Lab
Georgia Institute of Technology
Atlanta, GA 303322/0280
santi@cc.gatech.edu

Enric Plaza
IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia
(Spain)
enric@iiia.csic.es

ABSTRACT

In this paper we will present an argumentation framework for learning agents (AMAL) designed for two purposes: (1) for joint deliberation, and (2) for learning from communication. The AMAL framework is completely based on learning from examples: the argument preference relation, the argument generation policy, and the counterargument generation policy are case-based techniques. For join deliberation, learning agents share their experience by forming a committee to decide upon some joint decision. We experimentally show that the argumentation among committees of agents improves both the individual and joint performance. For learning from communication, an agent engages into arguing with other agents in order to contrast its individual hypotheses and receive counterexamples; the argumentation process improves their learning scope and individual performance.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems, Intelligent Agents*

Keywords

multi-agent learning, argumentation, case-based reasoning

1. INTRODUCTION

Argumentation frameworks for multi-agent systems can be used for different purposes like joint deliberation, persuasion, negotiation, and conflict resolution. In this paper we will present an argumentation framework for learning agents, and show that it can be used for two purposes: (1) joint deliberation, and (2) learning from communication.

Argumentation-based joint deliberation involves discussion over the outcome of a particular situation or the appropriate course of action for a particular situation. Learning agents are capable of learning from experience, in the sense that past examples (situations and their outcomes) are used to predict the outcome for the situation at hand. However, since individual agents experience may be limited, individual knowledge and prediction accuracy is also limited. Thus, learning agents that are capable of arguing their individual predictions with other agents may reach better prediction accuracy after such an argumentation process.

Most existing argumentation frameworks for multi-agent systems are based on deductive logic or some other deductive logic formalism specifically designed to support argumentation, such as default logic [3]). Usually, an argument is seen as a logical statement, while a counterargument is an argument offered in opposition to another argument [4, 13]; agents use a preference relation to resolve conflicting arguments. However, logic-based argumentation frameworks assume agents with preloaded knowledge and preference relation. In this paper, we focus on an *Argumentation-based Multi-Agent Learning* (AMAL) framework where both knowledge and preference relation are learned from experience. Thus, we consider a scenario with agents that (1) work in the same domain using a shared ontology, (2) are capable of learning from examples, and (3) communicate using an argumentative framework.

Having learning capabilities allows agents effectively use a specific form of counterargument, namely the use of *counterexamples*. Counterexamples offer the possibility of agents learning *during* the argumentation process. Moreover, learning agents allow techniques that use learnt experience to generate adequate arguments and counterarguments. Specifically, we will need to address two issues: (1) how to define a technique to generate arguments and counterarguments from examples, and (2)how to define a preference relation over two conflicting arguments that have been induced from examples.

This paper presents a case-based approach to address both issues. The agents use case-based reasoning (CBR) [1] to learn from past cases (where a case is a situation and its outcome) in order to predict the outcome of a new situation. We propose an argumentation protocol inside the AMAL framework at supports agents in reaching a joint prediction over a specific situation or problem — moreover, the reasoning needed to support the argumentation process will also be based on cases. In particular, we present two *case-based measures*, one for generating the arguments and counterarguments adequate to a particular situation and another for determining preference relation among arguments. Finally, we evaluate (1) if argumentation between learning agents can produce a joint prediction that improves over individual learning performance and (2) if learning from the counterexamples conveyed during the argumentation process increases the individual performance with precisely those cases being used while arguing among them.

The paper is structured as follows. Section 2 discusses the relation among argumentation, collaboration and learning. Then Sec-

tion 3 introduces our multi-agent CBR ($\mathcal{M}$AC) framework and the notion of justified prediction. After that, Section 4 formally defines our argumentation framework. Sections 5 and 6 present our case-based preference relation and argument generation policies respectively. Later, Section 7 presents the argumentation protocol in our AMAL framework. After that, Section 8 presents an exemplification of the argumentation framework. Finally, Section 9 presents an empirical evaluation of our two main hypotheses. The paper closes with related work and conclusions sections.

2. ARGUMENTATION, COLLABORATION AND LEARNING

Both learning and collaboration are ways in which an agent can improve individual performance. In fact, there is a clear parallelism between learning and collaboration in multi-agent systems, since both are ways in which agents can deal with their shortcomings. Let us show which are the main motivations that an agent can have to learn or to collaborate.

- Motivations to learn:
  - Increase quality of prediction,
  - Increase efficiency,
  - Increase the range of solvable problems.
- Motivations to collaborate:
  - Increase quality of prediction,
  - Increase efficiency,
  - Increase the range of solvable problems,
  - Increase the range of accessible resources.

Looking at the above lists of motivation, we can easily see that learning and collaboration are very related in multi-agent systems. In fact, with the exception of the last item in the motivations to collaborate list, they are two extremes of a continuum of strategies to improve performance. An agent may choose to increase performance by learning, by collaborating, or by finding an intermediate point that combines learning and collaboration in order to improve performance.

In this paper we will propose AMAL, an argumentation framework for learning agents, and will also also show how AMAL can be used both for learning from communication and for solving problems in a collaborative way:

- Agents can solve problems in a collaborative way via engaging an argumentation process about the prediction for the situation at hand. Using this collaboration, the prediction can be done in a more informed way, since the information known by several agents has been taken into account.
- Agents can also learn from communication with other agents by engaging an argumentation process. Agents that engage in such argumentation processes can learn from the arguments and counterexamples received from other agents, and use this information for predicting the outcomes of future situations.

In the rest of this paper we will propose an argumentation framework and show how it can be used both for learning and for solving problems in a collaborative way.

3. MULTI-AGENT CBR SYSTEMS

A *Multi-Agent Case Based Reasoning System* ($\mathcal{M}$AC) $\mathcal{M} = \{(A_1, C_1), ..., (A_n, C_n)\}$ is a multi-agent system composed of $\mathcal{A} = \{A_i, ..., A_n\}$, a set of CBR agents, where each agent $A_i \in \mathcal{A}$ possesses an individual case base C_i. Each individual agent A_i in a $\mathcal{M}$AC is completely autonomous and each agent A_i has access only to its individual and private case base C_i. A case base $C_i = \{c_1, ..., c_m\}$ is a collection of cases. Agents in a $\mathcal{M}$AC system are able to individually solve problems, but they can also collaborate with other agents to solve problems.

In this framework, we will restrict ourselves to analytical tasks, i.e. tasks like classification, where the solution of a problem is achieved by selecting a solution class from an enumerated set of solution classes. In the following we will note the set of all the solution classes by $\mathcal{S} = \{S_1, ..., S_K\}$. Therefore, a *case* $c = \langle P, S \rangle$ is a tuple containing a case description P and a solution class $S \in \mathcal{S}$. In the following, we will use the terms *problem* and *case description* indistinctly. Moreover, we will use the dot notation to refer to elements inside a tuple; e.g., to refer to the solution class of a case c, we will write $c.S$.

Therefore, we say a group of agents perform *joint deliberation*, when they collaborate to find a joint solution by means of an argumentation process. However, in order to do so, an agent has to be able to *justify* its prediction to the other agents (i.e. generate an argument for its predicted solution that can be examined and critiqued by the other agents). The next section addresses this issue.

3.1 Justified Predictions

Both expert systems and CBR systems may have an explanation component [14] in charge of justifying why the system has provided 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.

Most of the existing work on explanation generation focuses on generating explanations to be provided to the user. However, in our approach we use explanations (or justifications) as a tool for improving communication and coordination among agents. We are interested in justifications since they can be used as arguments. For that purpose, we will benefit from the ability of some machine learning methods to provide justifications.

A *justification* built by a CBR method after determining that the solution of a particular problem P was S_k is a description that contains the relevant information from the problem P that the CBR method has considered to predict S_k as the solution of P. In particular, CBR methods work by retrieving similar cases to the problem at hand, and then reusing their solutions for the current problem, expecting that since the problem and the cases are similar, the solutions will also be similar. Thus, if a CBR method has retrieved a set of cases $C_1, ..., C_n$ to solve a particular problem P the justification built will contain the relevant information from the problem P that made the CBR system retrieve that particular set of cases, i.e. it will contain the relevant information that P and $C_1, ..., C_n$ have in common.

For example, Figure 1 shows a justification build by a CBR system for a toy problem (in the following sections we will show justifications for real problems). In the figure, a problem has two attributes (*Traffic_light*, and *Cars_passing*), the retrieval mechanism of the CBR system notices that by considering only the attribute *Traffic_light*, it can retrieve two cases that predict the same solution: *wait*. Thus, since only this attribute has been used, it is the only one appearing in the justification. The values of the rest of attributes are irrelevant, since whatever their value the solution class would have been the same.

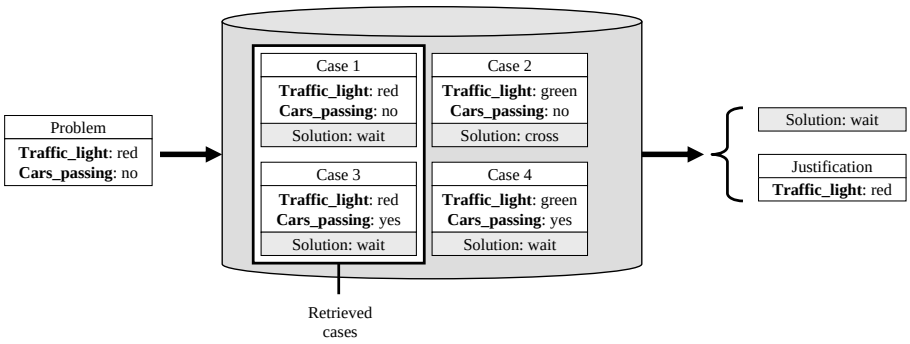

Figure 1: An example of justification generation in a CBR system. Notice that, since the only relevant feature to decide is *Traffic_light* (the only one used to retrieve cases), it is the only one appearing in the justification.

In general, the meaning of a justification is that all (or most of) the cases in the case base of an agent that satisfy the justification (i.e. all the cases that are *subsumed* by the justification) belong to the predicted solution class. In the rest of the paper, we will use $\sqsubseteq$ to denote the subsumption relation. In our work, we use LID [2], a CBR method capable of building symbolic justifications such as the one exemplified in Figure 1. When an agent provides a justification for a prediction, the agent generates a *justified prediction*:

DEFINITION 3.1. *A* Justified Prediction *is a tuple* $J = \langle A, P, S, D\rangle$ *where agent* A *considers* S *the correct solution for problem* P*, and that prediction is justified a symbolic description* D *such that* $J.D \sqsubseteq J.P$.

Justifications can have many uses for CBR systems [8, 9]. In this paper, we are going to use justifications as arguments, in order to allow learning agents to engage in argumentation processes.

4. ARGUMENTS AND COUNTERARGUMENTS

For our purposes an *argument* α generated by an agent A is composed of a statement S and some evidence D supporting S as correct. In the remainder of this section we will see how this general definition of argument can be instantiated in specific kind of arguments that the agents can generate. In the context of $\mathcal{M}$AC systems, agents argue about predictions for new problems and can provide two kinds of information: a) specific cases $\langle P, S\rangle$, and b) justified predictions: $\langle A, P, S, D\rangle$. Using this information, we can define three types of arguments: justified predictions, counterarguments, and counterexamples.

A *justified prediction* α is generated by an agent A_i to argue that A_i believes that the correct solution for a given problem P is $\alpha.S$, and the evidence provided is the justification $\alpha.D$. In the example depicted in Figure 1, an agent A_i may generate the argument $\alpha = \langle A_i, P, \textit{Wait}, (\textit{Traffic\_light} = \textit{red})\rangle$, meaning that the agent A_i believes that the correct solution for P is *Wait* because the attribute *Traffic_light* equals *red*.

A *counterargument* β is an argument offered in opposition to another argument α. In our framework, a counterargument consists of a justified prediction $\langle A_j, P, S', D'\rangle$ generated by an agent A_j with the intention to rebut an argument α generated by another agent A_i, that endorses a solution class S' different from that of $\alpha.S$ for the problem at hand and justifies this with a justification D'. In the example in Figure 1, if an agent generates the argument $\alpha = \langle A_i, P, \textit{Walk}, (\textit{Cars\_passing} = \textit{no})\rangle$, an agent that thinks that the correct solution is *Wait* might answer with the counterargument $\beta = \langle A_j, P, \textit{Wait}, (\textit{Cars\_passing} = \textit{no} \wedge \textit{Traffic\_light} = \textit{red})\rangle$, meaning that, although there are no cars passing, the traffic light is red, and the street cannot be crossed.

A *counterexample* c is a case that contradicts an argument α. Thus a counterexample is also a counterargument, one that states that a specific argument α is not always true, and the evidence provided is the case c. Specifically, for a case c to be a counterexample of an argument α, the following conditions have to be met: $\alpha.D \sqsubseteq c$ and $\alpha.S \neq c.S$, i.e. the case must satisfy the justification $\alpha.D$ and the solution of c must be different than the predicted by α.

By exchanging arguments and counterarguments (including counterexamples), agents can argue about the correct solution of a given problem, i.e. they can engage a joint deliberation process. However, in order to do so, they need a specific interaction protocol, a preference relation between contradicting arguments, and a decision policy to generate counterarguments (including counterexamples). In the following sections we will present these elements.

5. PREFERENCE RELATION

A specific argument provided by an agent might not be consistent with the information known to other agents (or even to some of the information known by the agent that has generated the justification due to noise in training data). For that reason, we are going to define a preference relation over contradicting justified predictions based on cases. Basically, we will define a *confidence* measure for each justified prediction (that takes into account the cases owned by each agent), and the justified prediction with the highest confidence will be the preferred one.

The idea behind case-based confidence is to count how many of the cases in an individual case base *endorse* a justified prediction, and how many of them are counterexamples of it. The more the endorsing cases, the higher the confidence; and the more the counterexamples, the lower the confidence. Specifically, to assess the confidence of a justified prediction α, an agent obtains the set of cases in its individual case base that are subsumed by $\alpha.D$. With them, an agent A_i obtains the Y (*aye*) and N (*nay*) values:

- $Y_\alpha^{A_i} = |\{c \in C_i |\ \alpha.D \sqsubseteq c.P \wedge \alpha.S = c.S\}|$ is the number of cases in the agent's case base *subsumed* by the justification $\alpha.D$ that belong to the solution class $\alpha.S$,
- $N_\alpha^{A_i} = |\{c \in C_i |\ \alpha.D \sqsubseteq c.P \wedge \alpha.S \neq c.S\}|$ is the number of cases in the agent's case base *subsumed* by justification $\alpha.D$ that do *not* belong to that solution class.

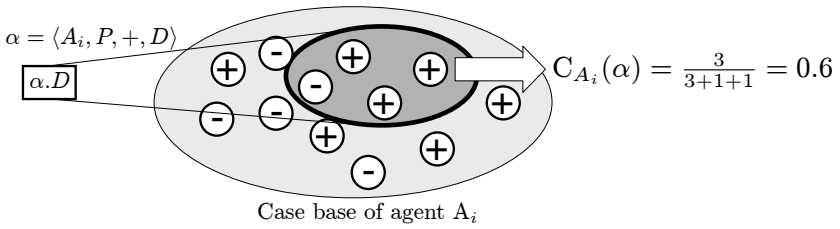

Figure 2: Confidence of arguments is evaluated by contrasting them against the case bases of the agents.

An agent estimates the confidence of an argument as:

$$\mathsf{C}_{A_i}(\alpha) = \frac{Y_\alpha^{A_i}}{1 + Y_\alpha^{A_i} + N_\alpha^{A_i}}$$

i.e. the confidence on a justified prediction is the number of endorsing cases divided by the number of endorsing cases plus counterexamples. Notice that we add 1 to the denominator, this is to avoid giving excessively high confidences to justified predictions whose confidence has been computed using a small number of cases. Notice that this correction follows the same idea than the Laplace correction to estimate probabilities. Figure 2 illustrates the individual evaluation of the confidence of an argument, in particular, three endorsing cases and one counterexample are found in the case base of agents A_i, giving an estimated confidence of 0.6

Moreover, we can also define the *joint confidence* of an argument α as the confidence computed using the cases present in the case bases of all the agents in the group:

$$\mathsf{C}(\alpha) = \frac{\sum_i Y_\alpha^{A_i}}{1 + \sum_i \left(Y_\alpha^{A_i} + N_\alpha^{A_i} \right)}$$

Notice that, to collaboratively compute the joint confidence, the agents only have to make public the aye and nay values locally computed for a given argument.

In our framework, agents use this joint confidence as the preference relation: a justified prediction α is preferred over another one β if $C(\alpha) \geq C(\beta)$.

6. GENERATION OF ARGUMENTS

In our framework, arguments are generated by the agents from cases, using learning methods. Any learning method able to provide a justified prediction can be used to generate arguments. For instance, decision trees and LID [2] are suitable learning methods. Specifically, in the experiments reported in this paper agents use LID. Thus, when an agent wants to generate an argument endorsing that a specific solution class is the correct solution for a problem P, it generates a justified prediction as explained in Section 3.1.

For instance, Figure 3 shows a real justification generated by LID after solving a problem P in the domain of marine sponges identification. In particular, Figure 3 shows how when an agent receives a new problem to solve (in this case, a new sponge to determine its order), the agent uses LID to generate an argument (consisting on a justified prediction) using the cases in the case base of the agent. The justification shown in Figure 3 can be interpreted saying that "the predicted solution is hadromerida because the smooth form of the megascleres of the spiculate skeleton of the sponge is of type tylostyle, the spikulate skeleton of the sponge has no uniform length, and there is no gemmules in the external features of the sponge". Thus, the argument generated will be $\alpha = \langle A_1, P, hadromerida, D_1 \rangle$.

6.1 Generation of Counterarguments

As previously stated, agents may try to rebut arguments by generating counterargument or by finding counterexamples. Let us explain how they can be generated.

An agent A_i wants to generate a counterargument β to rebut an argument α when α is in contradiction with the local case base of A_i. Moreover, while generating such counterargument β, A_i expects that β is preferred over α. For that purpose, we will present a specific policy to generate counterarguments based on the *specificity* criterion [10].

The specificity criterion is widely used in deductive frameworks for argumentation, and states that between two conflicting arguments, the most specific should be preferred since it is, in principle, more informed. Thus, counterarguments generated based on the specificity criterion are expected to be preferable (since they are more informed) to the arguments they try to rebut. However, there is no guarantee that such counterarguments will always win, since, as we have stated in Section 5, agents in our framework use a preference relation based on joint confidence. Moreover, one may think that it would be better that the agents generate counterarguments based on the joint confidence preference relation; however it is not obvious how to generate counterarguments based on joint confidence in an efficient way, since collaboration is required in order to evaluate joint confidence. Thus, the agent generating the counterargument should constantly communicate with the other agents at each step of the induction algorithm used to generate counterarguments (presently one of our future research lines).

Thus, in our framework, when an agent wants to generate a counterargument β to an argument α, β has to be more specific than α (i.e. $\alpha.D \sqsubset \beta.D$).

The generation of counterarguments using the specificity criterion imposes some restrictions over the learning method, although LID or ID3 can be easily adapted for this task. For instance, LID is an algorithm that generates a description starting from scratch and heuristically adding features to that term. Thus, at every step, the description is made more specific than in the previous step, and the number of cases that are subsumed by that description is reduced. When the description covers only (or almost only) cases of a single solution class LID terminates and predicts that solution class. To generate a counterargument to an argument α LID just has to use as starting point the description $\alpha.D$ instead of starting from scratch. In this way, the justification provided by LID will always be subsumed by $\alpha.D$, and thus the resulting counterargument will be more specific than α. However, notice that LID may sometimes not be able to generate counterarguments, since LID may not be able to specialize the description $\alpha.D$ any further, or because the agent A_i has no case inC_i that is subsumed by $\alpha.D$. Figure 4 shows how an agent A_2 that disagreed with the argument shown in Figure 3, generates a counterargument using LID. Moreover, Figure 4 shows the generation of a counterargument β_2^1 for the argument α_1^0 (in Figure 3) that is a specialization of α_1^0.

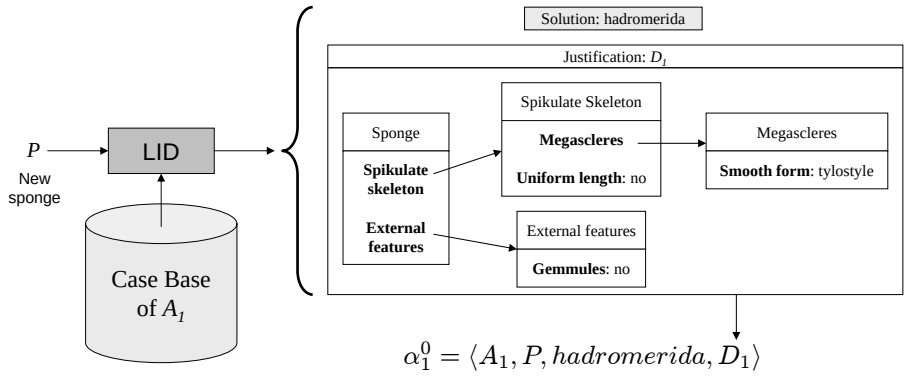

Figure 3: Example of a real justification generated by LID in the marine sponges data set.

Specifically, in our experiments, when an agent A_i wants to rebut an argument α, uses the following policy:

1. Agent A_i uses LID to try to find a counterargument β more specific than α; if found, β is sent to the other agent as a counterargument of α.
2. If not found, then A_i searches for a counterexample $c \in C_i$ of α. If a case c is found, then c is sent to the other agent as a counterexample of α.
3. If no counterexamples are found, then A_i cannot rebut the argument α.

7. ARGUMENTATION-BASED MULTI-AGENT LEARNING

The interaction protocol of AMAL allows a group of agents A_1, ..., A_n to deliberate about the correct solution of a problem P by means of an argumentation process. If the argumentation process arrives to a consensual solution, the joint deliberation ends; otherwise a weighted vote is used to determine the joint solution. Moreover, AMAL also allows the agents to learn from the counterexamples received from other agents.

The AMAL protocol consists on a series of rounds. In the initial round, each agent states which is its individual prediction for P. Then, at each round an agent can try to rebut the prediction made by any of the other agents. The protocol uses a token passing mechanism so that agents (one at a time) can send counterarguments or counterexamples if they disagree with the prediction made by any other agent. Specifically, each agent is allowed to send one counterargument or counterexample each time he gets the token (notice that this restriction is just to simplify the protocol, and that it does not restrict the number of counterargument an agent can sent, since they can be delayed for subsequent rounds). When an agent receives a counterargument or counterexample, it informs the other agents if it accepts the counterargument (and changes its prediction) or not. Moreover, agents have also the opportunity to answer to counterarguments when they receive the token, by trying to generate a counterargument to the counterargument.

When all the agents have had the token once, the token returns to the first agent, and so on. If at any time in the protocol, all the agents agree or during the last n rounds no agent has generated any counterargument, the protocol ends. Moreover, if at the end of the argumentation the agents have not reached an agreement, then a voting mechanism that uses the confidence of each prediction as weights is used to decide the final solution (Thus, AMAL follows the same mechanism as human committees, first each individual member of a committee exposes his arguments and discuses those of the other members (joint deliberation), and if no consensus is reached, then a voting mechanism is required).

At each iteration, agents can use the following performatives:

- $assert(\alpha)$: the justified prediction held during the next round will be α. An agent can only hold a single prediction at each round, thus is multiple asserts are send, only the last one is considered as the currently held prediction.
- $rebut(\beta, \alpha)$: the agent has found a counterargument β to the prediction α.

We will define $H_t = \langle \alpha_1^t, ..., \alpha_n^t \rangle$ as the predictions that each of the n agents hold at a round t. Moreover, we will also define $contradict(\alpha_i^t) = \{\alpha \in H_t | \alpha.S \neq \alpha_i^t.S\}$ as the set of contradicting arguments for an agent A_i in a round t, i.e. the set of arguments at round t that support a different solution class than α_i^t.

The protocol is initiated because one of the agents receives a problem P to be solved. After that, the agent informs all the other agents about the problem P to solve, and the protocol starts:

1. At round $t = 0$, each one of the agents individually solves P, and builds a justified prediction using its own CBR method. Then, each agent A_i sends the performative $assert(\alpha_i^0)$ to the other agents. Thus, the agents know $H_0 = \langle \alpha_i^0, ..., \alpha_n^0 \rangle$. Once all the predictions have been sent the token is given to the first agent A_1.
2. At each round t (other than 0), the agents check whether their arguments in H_t agree. If they do, the protocol moves to step 5. Moreover, if during the last n rounds no agent has sent any counterexample or counterargument, the protocol also moves to step 5. Otherwise, the agent A_i owner of the token tries to generate a counterargument for each of the opposing arguments in $contradict(\alpha_i^t) \subseteq H_t$ (see Section 6.1). Then, the counterargument β_i^t against the prediction α_j^t with the lowest confidence $C(\alpha_j^t)$ is selected (since α_j^t is the prediction more likely to be successfully rebutted).
    - If β_i^t is a counterargument, then, A_i locally compares α_i^t with β_i^t by assessing their confidence against its individual case base C_i (see Section 5) (notice that A_i is comparing its previous argument with the counterargument that A_i itself has just generated and that is about

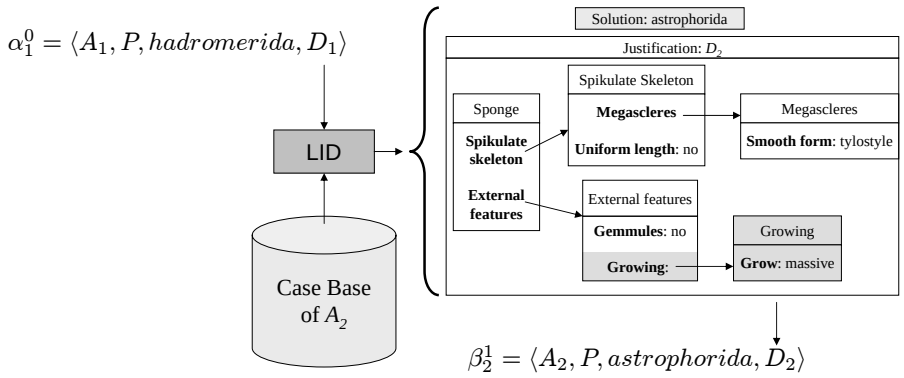

Figure 4: Generation of a counterargument using LID in the sponges data set.

to send to A_j). If $C_{A_i}(\beta_i^t) > C_{A_i}(\alpha_i^t)$, then A_i considers that β_i^t is stronger than its previous argument, changes its argument to β_i^t by sending $assert(\beta_i^t)$ to the rest of the agents (the intuition behind this is that since a counterargument is also an argument, A_i checks if the newly counterargument is a better argument than the one he was previously holding) and $rebut(\beta_i^t, \alpha_j^t)$ to A_j. Otherwise (i.e. $C_{A_i}(\beta_i^t) \leq C_{A_i}(\alpha_i^t)$), A_i will send only $rebut(\beta_i^t, \alpha_j^t)$ to A_j. In any of the two situations the protocol moves to step 3.

- If β_i^t is a counterexample c, then A_i sends $rebut(c, \alpha_j^t)$ to A_j. The protocol moves to step 4.
- If A_i cannot generate any counterargument or counterexample, the token is sent to the next agent, a new round $t+1$ starts, and the protocol moves to state 2.

3. The agent A_j that has received the counterargument β_i^t, locally compares it against its own argument, α_j^t, by locally assessing their confidence. If $C_{A_j}(\beta_i^t) > C_{A_j}(\alpha_j^t)$, then A_j will accept the counterargument as stronger than its own argument, and it will send $assert(\beta_i^t)$ to the other agents. Otherwise (i.e. $C_{A_j}(\beta_i^t) \leq C_{A_j}(\alpha_j^t)$), A_j will not accept the counterargument, and will inform the other agents accordingly. Any of the two situations start a new round $t+1$, A_i sends the token to the next agent, and the protocol moves back to state 2.

4. The agent A_j that has received the counterexample c retains it into its case base and generates a new argument α_j^{t+1} that takes into account c, and informs the rest of the agents by sending $assert(\alpha_j^{t+1})$ to all of them. Then, A_i sends the token to the next agent, a new round $t+1$ starts, and the protocol moves back to step 2.

5. The protocol ends yielding a joint prediction, as follows: if the arguments in H_t agree then their prediction is the joint prediction, otherwise a voting mechanism is used to decide the joint prediction. The voting mechanism uses the joint confidence measure as the voting weights, as follows:

$$S = \arg\max_{S_k \in \mathcal{S}} \sum_{\alpha_i \in H_t | \alpha_i.S = S_k} C(\alpha_i)$$

Moreover, in order to avoid infinite iterations, if an agent sends twice the same argument or counterargument to the same agent, the message is not considered.

8. EXEMPLIFICATION

Let us consider a system composed of three agents A_1, A_2 and A_3. One of the agents, A_1 receives a problem P to solve, and decides to use AMAL to solve it. For that reason, invites A_2 and A_3 to take part in the argumentation process. They accept the invitation, and the argumentation protocol starts.

Initially, each agent generates its individual prediction for P, and broadcasts it to the other agents. Thus, all of them can compute $H_0 = \langle \alpha_1^0, \alpha_2^0, \alpha_3^0 \rangle$. In particular, in this example:

- $\alpha_1^0 = \langle A_1, P, hadromerida, D_1 \rangle$
- $\alpha_2^0 = \langle A_2, P, astrophorida, D_2 \rangle$
- $\alpha_3^0 = \langle A_3, P, axinellida, D_3 \rangle$

A_1 starts owning the token and tries to generate counterarguments for α_2^0 and α_3^0, but does not succeed, however it has one counterexample c_{13} for α_3^0. Thus, A_1 sends the the message $rebut(c_{13}, \alpha_3^0)$ to A_3. A_3 incorporates c_{13} into its case base and tries to solve the problem P again, now taking c_{13} into consideration. A_3 comes up with the justified prediction $\alpha_3^1 = \langle A_3, P, hadromerida, D_4 \rangle$, and broadcasts it to the rest of the agents with the message $assert(\alpha_3^1)$. Thus, all of them know the new $H_1 = \langle \alpha_1^0, \alpha_2^0, \alpha_3^1 \rangle$.

Round 1 starts and A_2 gets the token. A_2 tries to generate counterarguments for α_1^0 and α_3^1 and only succeeds to generate a counterargument $\beta_2^1 = \langle A_2, P, astrophorida, D_5 \rangle$ against α_3^1. The counterargument is sent to A_3 with the message $rebut(\beta_2^1, \alpha_3^1)$. Agent A_3 receives the counterargument and assesses its local confidence. The result is that the individual confidence of the counterargument β_2^1 is lower than the local confidence of α_3^1. Therefore, A_3 does not accept the counterargument, and thus $H_2 = \langle \alpha_1^0, \alpha_2^0, \alpha_3^1 \rangle$.

Round 2 starts and A_3 gets the token. A_3 generates a counterargument $\beta_3^2 = \langle A_3, P, hadromerida, D_6 \rangle$ for α_2^0 and sends it to A_2 with the message $rebut(\beta_3^2, \alpha_2^0)$. Agent A_2 receives the counterargument and assesses its local confidence. The result is that the local confidence of the counterargument β_3^2 is higher than the local confidence of α_2^0. Therefore, A_2 accepts the counterargument and informs the rest of the agents with the message $assert(\beta_3^2)$. After that, $H_3 = \langle \alpha_1^0, \beta_3^2, \alpha_3^1 \rangle$.

At Round 3, since all the agents agree (all the justified predictions in H_3 predict $hadromerida$ as the solution class) The protocol ends, and A_1 (the agent that received the problem) considers $hadromerida$ as the joint solution for the problem P.

9. EXPERIMENTAL EVALUATION

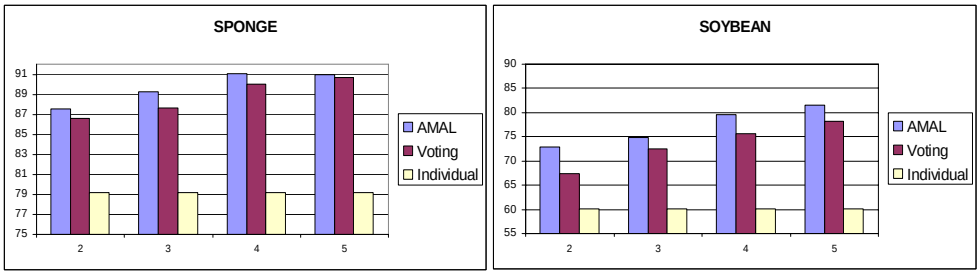

Figure 5: Individual and joint accuracy for 2 to 5 agents.

In this section we empirically evaluate the AMAL argumentation framework. We have made experiments in two different data sets: *soybean* (from the UCI machine learning repository) and *sponge* (a relational data set). The soybean data set has 307 examples and 19 solution classes, while the sponge data set has 280 examples and 3 solution classes. In an experimental run, the data set is divided in 2 sets: the training set and the test set. The training set examples are distributed among 5 different agents without replication, i.e. there is no example shared by two agents. In the testing stage, problems in the test set arrive randomly to one of the agents, and their goal is to predict the correct solution.

The experiments are designed to test two hypotheses: (H1) that argumentation is a useful framework for joint deliberation and can improve over other typical methods such as voting; and (H2) that learning from communication improves the individual performance of a learning agent participating in an argumentation process. Moreover, we also expect that the improvement achieved from argumentation will increase as the number of agents participating in the argumentation increases (since more information will be taken into account).

Concerning H1 (argumentation is a useful framework for joint deliberation), we ran 4 experiments, using 2, 3, 4, and 5 agents respectively (in all experiments each agent has a 20% of the training data, since the training is always distributed among 5 agents).

Figure 5 shows the result of those experiments in the sponge and soybean data sets. Classification accuracy is plotted in the vertical axis, and in the horizontal axis the number of agents that took part in the argumentation processes is shown. For each number of agents, three bars are shown: *individual*, *Voting*, and AMAL. The individual bar shows the average accuracy of individual agents predictions; the voting bar shows the average accuracy of the joint prediction achieved by voting but without any argumentation; and finally the AMAL bar shows the average accuracy of the joint prediction using argumentation. The results shown are the average of 5 10-fold cross validation runs.

Figure 5 shows that collaboration (voting and AMAL) outperforms individual problem solving. Moreover, as we expected, the accuracy improves as more agents collaborate, since more information is taken into account. We can also see that AMAL always outperforms standard voting, proving that joint decisions are based on better information as provided by the argumentation process.

For instance, the joint accuracy for 2 agents in the sponge data set is of 87.57% for AMAL and 86.57% for voting (while individual accuracy is just 80.07%). Moreover, the improvement achieved by AMAL over Voting is even larger in the soybean data set. The reason is that the soybean data set is more "difficult" (in the sense that agents need more data to produce good predictions). These experimental results show that AMAL effectively exploits the opportunity for improvement: the accuracy is higher only because more agents have changed their opinion during argumentation (otherwise they would achieve the same result as Voting).

Concerning H2 (learning from communication in argumentation processes improves individual prediction), we ran the following experiment: initially, we distributed a 25% of the training set among the five agents; after that, the rest of the cases in the training set is sent to the agents one by one; when an agent receives a new training case, it has several options: the agent can discard it, the agent can retain it, or the agent can use it for engaging an argumentation process. Figure 6 shows the result of that experiment for the two data sets. Figure 6 contains three plots, where NL (not learning) shows accuracy of an agent with no learning at all; L (learning), shows the evolution of the individual classification accuracy when agents learn by retaining the training cases they individually receive (notice that when all the training cases have been retained at 100%, the accuracy should be equal to that of Figure 5 for individual agents); and finally LFC (learning from communication) shows the evolution of the individual classification accuracy of learning agents that also learn by retaining those counterexamples received during argumentation (i.e. they learn both from training examples and counterexamples).

Figure 6 shows that if an agent A_i learns also from communication, A_i can significantly improve its individual performance with just a small number of additional cases (those selected as relevant counterexamples for A_i during argumentation). For instance, in the soybean data set, individual agents have achieved an accuracy of 70.62% when they also learn from communication versus an accuracy of 59.93% when they only learn from their individual experience. The number of cases learnt from communication depends on the properties of the data set: in the sponges data set, agents have retained only very few additional cases, and significantly improved individual accuracy; namely they retain 59.96 cases in average (compared to the 50.4 cases retained if they do not learn from communication). In the soybean data set more counterexamples are learnt to significantly improve individual accuracy, namely they retain 87.16 cases in average (compared to 55.27 cases retained if they do not learn from communication). Finally, the fact that both data sets show a significant improvement points out the adaptive nature of the argumentation-based approach to learning from communication: the useful cases are selected as counterexamples (and no more than those needed), and they have the intended effect.

10. RELATED WORK

Concerning CBR in a multi-agent setting, the first research was on "negotiated case retrieval" [11] among groups of agents. Our work on multi-agent case-based learning started in 1999 [6]; later Mc Ginty and Smyth [7] presented a multi-agent collaborative CBR approach (CCBR) for planning. Finally, another interesting approach is *multi-case-base reasoning* (MCBR) [5], that deals with

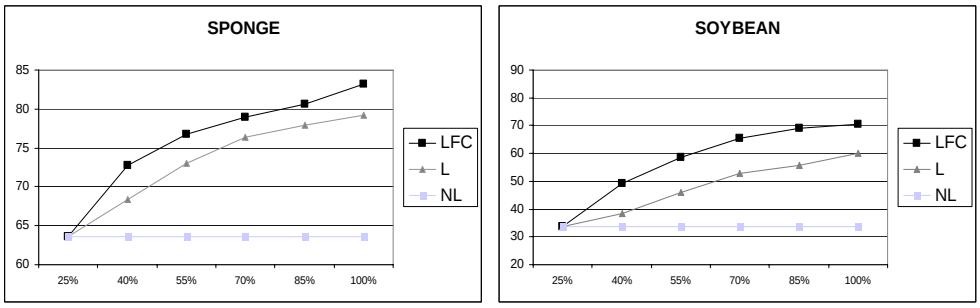

Figure 6: Learning from communication resulting from argumentation in a system composed of 5 agents.

distributed systems where there are several case bases available for the same task and addresses the problems of cross-case base adaptation. The main difference is that our $\mathcal{M}$AC approach is a way to distribute the *Reuse* process of CBR (using a voting system) while *Retrieve* is performed individually by each agent; the other multi-agent CBR approaches, however, focus on distributing the *Retrieve* process.

Research on MAS argumentation focus on several issues like a) logics, protocols and languages that support argumentation, b) argument selection and c) argument interpretation. Approaches for logic and languages that support argumentation include defeasible logic [4] and BDI models [13]. Although argument selection is a key aspect of automated argumentation (see [12] and [13]), most research has been focused on preference relations among arguments. In our framework we have addressed both argument selection and preference relations using a case-based approach.

11. CONCLUSIONS AND FUTURE WORK

In this paper we have presented an argumentation-based framework for multi-agent learning. Specifically, we have presented AMAL, a framework that allows a group of learning agents to argue about the solution of a given problem and we have shown how the learning capabilities can be used to generate arguments and counterarguments. The experimental evaluation shows that the increased amount of information provided to the agents by the argumentation process increases their predictive accuracy, and specially when an adequate number of agents take part in the argumentation.

The main contributions of this work are: a) an argumentation framework for learning agents; b) a case-based preference relation over arguments, based on computing an overall confidence estimation of arguments; c) a case-based policy to generate counterarguments and select counterexamples; and d) an argumentation-based approach for learning from communication.

Finally, in the experiments presented here a learning agent would retain all counterexamples submitted by the other agent; however, this is a very simple *case retention policy*, and we will like to experiment with more informed policies — with the goal that individual learning agents could significantly improve using only a small set of cases proposed by other agents. Finally, our approach is focused on lazy learning, and future works aims at incorporating eager inductive learning inside the argumentative framework for learning from communication.

12. REFERENCES

[1] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39–59, 1994.

[2] E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In *ECML'2001*, pages 13–24, 2001.

[3] Gerhard Brewka. Dynamic argument systems: A formal model of argumentation processes based on situation calculus. *Journal of Logic and Computation*, 11(2):257–282, 2001.

[4] Carlos I. Chesñevar and Guillermo R. Simari. Formalizing Defeasible Argumentation using Labelled Deductive Systems. *Journal of Computer Science & Technology*, 1(4):18–33, 2000.

[5] D. Leake and R. Sooriamurthi. Automatically selecting strategies for multi-case-base reasoning. In S. Craw and A. Preece, editors, *ECCBR'2002*, pages 204–219, Berlin, 2002. Springer Verlag.

[6] Francisco J. Martín, Enric Plaza, and Josep-Lluis Arcos. Knowledge and experience reuse through communications among competent (peer) agents. *International Journal of Software Engineering and Knowledge Engineering*, 9(3):319–341, 1999.

[7] Lorraine McGinty and Barry Smyth. Collaborative case-based reasoning: Applications in personalized route planning. In I. Watson and Q. Yang, editors, *ICCBR*, number 2080 in LNAI, pages 362–376. Springer-Verlag, 2001.

[8] Santi Ontañón and Enric Plaza. Justification-based multiagent learning. In *ICML'2003*, pages 576–583. Morgan Kaufmann, 2003.

[9] Enric Plaza, Eva Armengol, and Santiago Ontañón. The explanatory power of symbolic similarity in case-based reasoning. *Artificial Intelligence Review*, 24(2):145–161, 2005.

[10] David Poole. On the comparison of theories: Preferring the most specific explanation. In *IJCAI-85*, pages 144–147, 1985.

[11] M V Nagendra Prassad, Victor R Lesser, and Susan Lander. Retrieval and reasoning in distributed case bases. Technical report, UMass Computer Science Department, 1995.

[12] K. Sycara S. Kraus and A. Evenchik. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence Journal*, 104:1–69, 1998.

[13] N. R. Jennings S. Parsons, C. Sierra. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8:261–292, 1998.

[14] Bruce A. Wooley. Explanation component of software systems. *ACM CrossRoads*, 5.1, 1998.