Inducing Domain Theory from Problem Solving in a Multi-agent System

Eloi Puertas and Eva Armengol
IIIA - Artificial Intelligence Research Institute,
CSIC - Spanish Council for Scientific Research,
Campus UAB, 08193 Bellaterra, Catalonia (Spain).
email: {eloi, eva}@iiia.csic.es,

Abstract.
The approach of in this paper tries to model the scenario of how an agent with poor domain experience could improve its problem solving behavior. In contrast to other approaches, we do not permit that agents exchange domain knowledge (neither cases nor domain theory). The agent with poor experience takes benefit of the problem solving behavior of other agents to improve its performance. Thus, it requests other agents for solving known problems and then induces one domain theory per requested agent. Finally, the agent achieves a higher accuracy in solving problems by his own using the induced domain theories.

1 Introduction

A multi-agent system (MAS) is composed of a collection of agents holding a set of properties [7] and also they are able to both coordinate and cooperate in order to achieve a goal. The introduction of learning capabilities into a MAS allows the improvement of the global problem solving behavior. Some approaches use inductive learning methods for concept learning on a MAS. The goal of concept learning is to induce a domain theory compatible with all positive and negative examples. Therefore, the goal of concept learning in MAS is to build an integrated domain theory compatible with the positive and negative examples of all agents. For instance, Davies and Edwards [4] propose an extension of the Version Space [8] method for concept learning in MAS. The goal is to integrate the version spaces of each agent in order to build a domain theory consistent with all the local domain knowledge. A similar idea is introduced by Brazdil and Torgo [3]. Here the authors consider that each agent is able to induce a domain theory and then all the individual domain theories are transferred to one of the agents who integrates them. In both approaches [4, 3], the integrated theory can be used by any of the agents belonging to the system to independently solving new problems. Notice that agents have to share domain information to build the integrated theory.

In our paper we propose that an agent can improve his accuracy by inducing a domain theory from the problem solving behavior of the other agents. Moreover, we do not permit the exchange of information among agents but only solutions of cases. Thus, one agent ask the others for solving problems and induces a domain theory taking into account only the descriptions of the problems solved correctly.

A different vision of the cooperation among several entities is that of *ensemble learning*. An ensemble is composed of several base classifiers (that use inductive learning methods), being each one of them capable of completely solving a problem from its own experience. Because the classifiers can provide different solutions for the same problem, the key issue of ensemble learning is how to aggregate different solutions proposed by different classifiers. Prodomidis et al. [13] distinguish four ways of performing such aggregation: voting, weighted voting, arbitrating, and combining. Through voting and weighted voting, the solution to a problem is proposed by the majority of base classifiers. Instead, arbitrating and combining perform a sort of meta-learning by learning either which classifiers are most preferred or combining the solutions respectively. Both, an arbiter and a combiner are trained on the predictions done by a particular set of classifiers.

Plaza and Ontañón [11] define a *commitee* as a set of agents, each one having his own experience and capable of completely solving new problems. Agents in a commitee can solve problems but they can also collaborate with others in order to improve their accuracy. The difference between this approach and the most common approaches to MAS learning is that all the agents in the system are capable of completely solving problems.

The approach introduced in this paper tries to model the scenario of how an agent belonging to a commitee could improve its problem solving behaviour. As in [11], we assume that agents of a commitee hold the following properties: 1) they are cooperative, i.e. they always will try to solve the problems; 2) the experience (case base) of each agent is different; 3) each agent is capable of completely solving a problem. Commonly, the improvement of the domain theory is done by acquiring new problems, but in our scenario the agents do not exchange neither cases (as in [10]) nor domain theory(as in [3, 4]). Instead, an agent asks to the others for the classification of some cases. Then from the proposed solutions, the first agent is able to induce a new domain theory that will be used in the future for solving new problems by his own.

On the other hand, when an agent has a little case base (i.e. poor experience), the domain theory resulting from the induction on these cases could also be poor. As a consequence, given a new problem, this agent either could not classify it because the induced domain theory is too specific, or the problem could have several solutions because the induced domain theory is too general. In both cases, the accuracy of the agent should not be satisfactory. With the approach we propose, that agent induces a domain theory from the problem solving behavior of each other agent. Thus, the accuracy of that agent is improved thanks to the aggregation of the solutions proposed by each domain theory.

In the next sections we explain in detail the approach. Section 2 describes the MAS scenario. Section 3 describes some experiments and discuss the results of them. Finally the paper closes with the conclusions and future work.

2 Description of the scenario

Let us suppose that an agent has not enough experience in problem solving, i.e., its case base contains few cases. The domain theory that could be induced from this case base is also poor. Therefore, the goal of this agent should be to acquire more cases to induce a better domain theory. Because we assume that this agent cannot exchange domain knowledge with other agents, the only possible situation is that one agent asks other agents to solve problems. Eventually, the agent could send to the requested agents only part of the problem description.

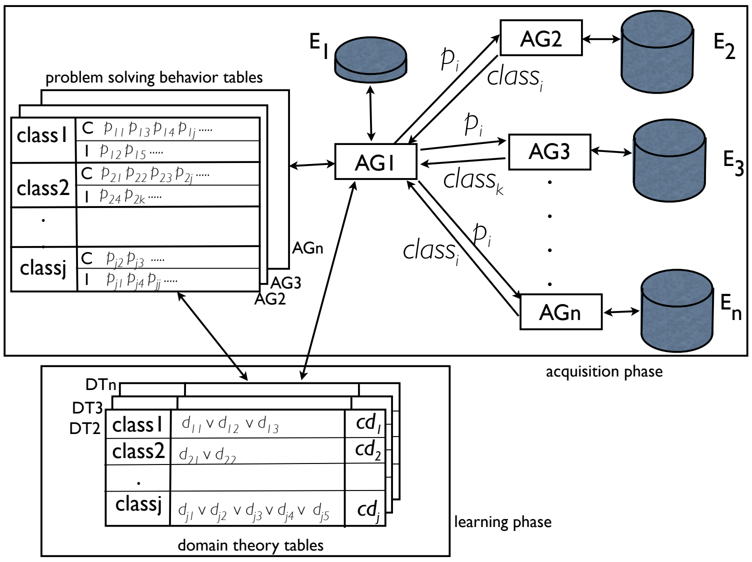
problem solving behavior tables
class1
C p_{11} p_{13} p_{14} p_{1j}.....
I p_{12} p_{15}.....
class2
C p_{21} p_{22} p_{23} p_{2j}.....
I p_{24} p_{2k}.....
classj
C p_{j2} p_{j3} .....
I p_{j1} p_{j4} p_{jj}.....
AGn
AG3
AG2
E1
AG1
AG2
AG3
AGn
E2
E3
En
p_i
$class_i$
p_i
$class_k$
p_i
$class_i$
acquisition phase
DTn
DT3
DT2
class1
d_{11} v d_{12} v d_{13}
cd_1
class2
d_{21} v d_{22}
cd_2
classj
d_{j1} v d_{j2} v d_{j3} v d_{j4} v d_{j5}
cd_j
domain theory tables
learning phase

Figure 1: Acquisition and learning phases.

The process we propose has three phases: *acquisition phase, learning phase,* and *problem solving phase*. Each agent AG_i owns a set of cases E_i. During the acquisition phase an agent with poor experience (say AG_1) sends to the other agents each case in E_1 and builds one problem solving behavior table for each requested agent. Then, during the learning phase, these tables are used by AG_1 to build a domain theory (DT_i) table for each agent AG_i. These tables contain, for each solution class $class_k$, a general description (commonly a disjunction of descriptions d_{ij}) with a certain degree of confidence cd_k. In the following we will explain these phases in detail.

Let M be a multi-agent system composed of n agents. Each agent AG_i has his own experience E_i. This experience is composed of a set of problems (cases) that the agent has solved in the past. A case c_i is a pair $(p_i, class_k)$ where p_i is the description of the problem and $class_k$ is the correct solution class where it belongs. Let us suppose now that AG_1 is the agent owning a small case base, so this agent should need to acquire more experience about the domain in order to induce a good domain theory. Therefore, AG_1 can initiate the *acquisition phase*.

During the acquisition phase, AG_1 asks the other agents $AG_2...AG_n$ to solve all the cases $c_i \in E_1$. These agents solve the problems using their own problem solving method and case base and, for each problem p_i, each agent AG_i proposes a solution class $class_j$. Notice that different agents can propose different solution classes according to their experience. Then, AG_1 compares the solution proposed by AG_i ($class_j$) and the correct solution class ($class_k$) and builds the *problem solving behavior* table (see Fig. 1). The problem solving behavior table of agent AG_i contains which problems p_j have been correctly solved (C in Fig. 1) and those incorrectly solved (I in Fig. 1). At the end of the acquisition phase, AG_1 has a problem solving behavior table for each agent AG_i. Each one of these tables contains, for each solution class, the problems of that solution class that AG_i has correctly solved and those that has incorrectly solved.

During the *learning phase*, for each agent AG_i, AG_1 uses the problem solving behavior table of that agent to induce a domain theory (DT_i). For each agent AG_i and each solution class $class_k$, AG_1 uses an inductive learning method to build a general description for that class. This method takes as examples the problems that the agent AG_i has correctly solved for $class_k$, and as negative examples the remaining ones (i.e. the problems in $class_k$ incorrectly solved and the problems belonging to the other classes). Moreover, the descriptions induced by AG_1 per class and agent are assigned a confidence degree cd_k (see Fig. 1). Given a class $class_k$ and an agent AG_i, the confidence degree that AG_1 assigns to the descriptions induced for $class_k$ is computed as follows:

$$cd_k = \frac{card(C_{AG_i})}{card(Total_{ik})}$$

where $card(C_{AG_i})$ is the number of problems in $class_k$ that AG_i has correctly solved; and $card(Total_{ik})$ is the number of cases that AG_i has (correctly or incorrectly) classified as belonging to $class_k$. Thus, when an agent AG_i correctly solves most problems of a solution class, the confidence degree that AG_1 has in AG_i for that class is high. Conversely, when the agent AG_i solves incorrectly most problems of a solution class the confidence degree for this class is low.

Because the domain theory DT_1 has been induced from few cases, the descriptions of the solution classes could not be accurate enough. For this reason, a new problem could satisfy descriptions of more than one solution class, i.e. it could be classified as belonging to several solution classes. As we explain below, the domain theories DT_i allows the use, during the problem solving phase, of an aggregation method that avoids multiple classifications of the new problems.

Notice that the domain theory DT_1 that AG_1 can induce from its own case base E_1 should not be exactly the same that the domain theories built from the problem solving behavior table of each agent AG_i. Let C_k be the set of cases in E_1 belonging to the $class_k$ solution class. The description for class $class_k$ in DT_1 has been built taking as examples the set C_k and as negative examples the set $E_1 - C_k$. Instead, the learning method induced each DT_i taking as examples only those problems in $class_k$ that have been correctly solved by AG_i and as negative examples the problems incorrectly solved in $class_k$ and the set $E_1 - C_k$. This means that DT_i should contain more specific descriptions than DT_1.

During the *problem solving phase*, agent AG_1 classifies new problems using the domain theory tables induced from the problem solving behavior tables of each agent. Let **DT**$_i$ be the domain theory table built by AG_1 from the problem solving behavior of agent AG_i; and $D_k = \{d_{kj}\}$ the descriptions induced for the solution class $class_k$ from the AG_i behavior. For each new problem p, AG_1 searches in the domain theory table of each agent for class descriptions covering p. Let $Cl = \{class_i | \exists d_{ij} \in D_i \textit{ such that } d_{ij} \textit{ covers } p\}$ be the multi-set of the classifications of p according to each domain theory DT_i, then AG_1 has to aggregate the solutions in Cl to find a classification for p. The aggregation method has three steps:

1. All the agents propose the same classification for p, i.e., $Cl = \{class_k\}$ for all the domain theories, in such situation AG_1 classifies p as belonging to $class_k$.

2. Most domain theories classify p as belonging to $class_k$, in such situation AG_1 classifies p as belonging to $class_k$ (*majority rule*).

3. If there is a tie situation among two or more classes, then AG_1 takes into account the confidence degree of the class descriptions of each domain theory. For each class $class_k$ in the tie situation, AG_1 computes confidence degree $CD(class_k)$ as follows:

$$CD(class_k) = \sum_{DT_i \in Correct_{DT}} cd_i$$

where $Correct_{DT}$ is the set of domain theories that proposed $class_k$ as the solution for p; and cd_i is the confidence degree of the domain theory DT_i for the solution class $class_k$. The winner class is $max\{CD(class_k)\}$, i.e., AG_1 classifies p as belonging to the solution class with higher confidence degree.

The aggregation method prefers the application of the majority rule before the use of the confidence degree. The reason is that, in principle, agent AG_1 has the same confidence on all the domain theories. Only when the majority rule produces a tie situation, AG_1 takes into account the confidence degree of each individual class description. In the future we plan to experiment with other aggregation methods.

The acquisition phase is an expensive process due to AG_1 asks to all other agents of the MAS. Nevertheless, this cost is compensated by the fact that during the problem solving phase, agent AG_1 autonomously solves new problems. In the future we plan to reduce the number of agents requested by AG_1.

3 Experimental Results

To implement the scenario described in the previous section, we need to determine both learning and problem solving methods used by the agents. In our experiments, we suppose that 1) all the agents use the LID method [2] for problem solving; and 2) the agent with little experience uses the INDIE method [1] to induce descriptions of the solution classes. The use of these methods is not a restriction since each agent could use any problem solving method. Also, the inductive learning method used to induce domain theory could be any of the usual methods (e.g. decision trees).

Lazy Induction of Descriptions (LID) is a lazy concept learning method for classification tasks in case-based reasoning (CBR). LID determines the most relevant attributes of a problem and searches in the case base for cases sharing these relevant attributes.

INDIE is a heuristic bottom-up inductive learning method that obtains a most specific generalization satisfied by a set of positive examples. Given a set of training examples $E = \{e_1, ..., e_m\}$ and a set of solution classes $C = \{class_1, ..., class_n\}$, the goal of INDIE is to obtain a discriminant description D_k for each solution class $class_k$.

3.1 Experiments

We run experiments on both the Car Evaluation and the Large Soybean databases from the UCI repository (www.ics.uci.edu/∼mlearn/MLRepository.html). Our multiagent system is composed of six agents, namely $AG_1 ... AG_6$. We considered that AG_1 is the one with little experience on the domain. Agents $AG_2 ... AG_6$ solve problems using the LID method and agent AG_1 uses INDIE to induce a domain theory. In the experiments we compared the accuracy of AG_1 using the domain theories induced from 1) its own case base; and 2) the problem solving behavior table of the other agents.

The Car Evaluation database contains 1728 examples representing descriptions of cars belonging to four solution classes: *unacc, acc, good* and *v-good*. There are not descriptions

Table 1: Experiments on Car Evaluation and Large soybean datasets. Columns $Card(E_1)$ and $Card(E_i)$ respectively shows the number of cases in the case base of AG_1 and other agents. The two right columns are the accuracy of AG_1 using both the domain theory from E_1, and the learned domain theory. In parenthesis there is the percentage of multiple solutions.

Dataset	Card(E_1)	Card(E_i)	own domain theory	learned domain theory
Car	50	301	61.86% (15.22 %)	75.61% (0.02%)
	100	291	64.64% (16.64 %)	78.67% (0.02 %)
Soybean	37	42	45.06 % (17.05 %)	52.69 % (0.37 %)
	50	39	54.38 % (15.64 %)	61.31 % (0.09 %)
	100	29	66.51 % (16.53 %)	68.43 % (0.05 %)

of cars with attributes having unknown value. The approach has been evaluated using 10-fold cross-validation, i.e. we randomly extracted 10% of the cases as the test set, while the rest were randomly distributed among all agents. We performed an experiment with AG_1 owning 50 cases and with each agent AG_i owning 301 cases. The results (see Table 1) show that the accuracy achieved by AG_1 using the learned domain theory (i.e., that induced from the problem solving behavior of the other agents) is 75.61% whereas the accuracy using its own domain theory is 61.86%. The results also show that with its own domain theory, AG_1 provides multiple solution classes for around 15% of the test cases. The percentage of multiple solutions is almost 0 when using the learned domain theory.

We conducted a second experiment with AG_1 having 100 cases and with each agent having 291 cases. Results are also similar to those of the first experiment: the accuracy of AG_1 using the learned domain theory is higher than using its own domain theory. Moreover, the percentage of multiple solutions also decreases using the learned domain theory. The analysis of these results shows that the increase of accuracy is a direct consequence of the decrement of multiple solutions. As explained before, the domain theory induced from the case base of AG_1 produces a high percentage of multiple solutions due to the overgeneralization of the induced descriptions. The aggregation of the solutions produced by each individual domain theory avoids this multiplicity.

In order to confirm the feasibility of the learned domain theory, we performed experiments with the Large Soybean database, whose characteristics are different to the Car database: the number of cases is smaller and there is a higher number of solution classes. The Large Soybean database contains 307 examples that can be classified as belonging to 19 solution classes. Moreover, the description of some examples have attributes with unknown values. Due to the small number of examples in this database, the evaluation has been done using 5-fold cross-validation, i.e. we randomly extracted 20% of cases as test set, while the rest were randomly distributed among the agents. We experimented with AG_1 having a case base closer in size to the case bases of the other agents. In particular, in the first experiment AG_1 has 37 cases and each AG_i has 42. In the second experiment AG_1 has 50 cases and each AG_i has 39 cases. Results of both experiments (see Table 1) show also that the accuracy of AG_1 using the learned domain theory is higher than using its own domain theory.

Finally, we conducted a third experiment with AG_1 whose case base is much larger than the other case bases. In particular, we considered that AG_1 has 100 cases and the case base of the other agents has 29 cases. In this experiment the accuracy using the learned theory (68.43%) is near the accuracy using its own domain theory (66.51%). Notice that now most of the multiple solutions of the own theory have been converted to failed classification in the learned theory.

3.2 Discussion

The results on both domains show that while the case base size of AG_1 is either smaller or closer to the case base size of the other agents, the accuracy using the learned domain theory is higher than using the own domain theory. This is consistent with the fact that inductive learning methods need many examples to induce good domain theories. Notice that accuracy on Soybean with AG_1 having 100 cases (3.5 times plus cases than the other agents) is similar using both the own domain theory and the learned domain theory. Therefore, our approach is feasible in situations where AG_1 has little experience. Nevertheless when its case base size is too small, this can prevent the learning of a domain theory from the other agents due to the fact that AG_1 has not enough cases to ask.

Most errors made by AG_1 using its own domain theory are due to the multiplicity of answers, i.e., to the agent not being able to determine a unique solution class for a problem. Instead, with the learned theory AG_1 takes benefit of the aggregation method to achieve a unique solution for each test case. This is a consequence of the *ensemble effect* that states that the resulting error of the combined predictions made by several independent classifiers is lower than the error of an individual classifier [5].

Because the problem solving method used by the agents could be changed, we also performed experiments considering that AG_1 uses the LID method on its own case base. The accuracy of LID on the Car database is 73% and 77% when AG_1 has respectively, 50 and 100 cases. These accuracies are higher than the accuracies obtained from its own domain theory (and closer to those of the learned theory). This result is consistent with the fact that case-based reasoning methods perform better than inductive methods with a small case base. Using LID on the soybean database, the accuracies of AG_1 are 53.07%, 58.08% and 74.61% with 37, 50 and 100 cases respectively. Notice that when AG_1 has 100 cases, the accuracy is higher using LID than using the learned theory. This is because AG_1 has the most of cases and, moreover it uses a CBR method.

4 Conclusions and Future Work

In this paper we propose the use of inductive learning techniques to improve the domain theory of one agent with poor experience on a domain. We assume that all the agents are able to completely solve problems of a domain and that they do not exchange neither cases nor domain theory. Our approach has three phases. The first one is the acquisition phase in which an agent asks to other agents for solving known problems and builds one problem solving behavior table for each agent. The second phase is the learning phase, in which the agent induces one domain theory from each problem solving behavior table. Finally, during the problem solving phase the agent uses the induced domain theories and an aggregation method to reach a solution for new problems. Notice that this MAS could be easily extended with other agents building the problem solving behavior table of each new agent and then inducing domain theory from it.

We run experiments comparing the accuracy of the agent with poor experience using the domain theory induced from its case base with the accuracy using the learned domain theory. Results show that our approach is feasible because supports the elimination of multiple solutions (failures) in classifying new problems taking benefit of the ensemble effect thanks to the aggregation method. This produces an increment of the agent accuracy.

As future work we plan to use different methods to aggregate the individual solutions. In particular, we could use a criterion based only on the confidence degree of the induced domain theories. Currently, the aggregation method does not take into account the solution proposed by the agent with less experience. In the future we could use meta-learning techniques, such as the combiners and arbiters, to reach the solution for a new problem and also to reduce the number of requested agents.

4.0.1 Acknowledgements

This work has been supported by the SAMAP project (TIC2002-04146-C05-01). The authors thank Santi Ontañón and José Antonio Rodríguez for their support in the elaboration of this paper.

References

[1] E. Armengol and E. Plaza. Bottom-up induction of feature terms. *Machine Learning*, 41(1):259–294, 2000.

[2] E. Armengol and E. Plaza. Lazy induction of descriptions for relational case-based learning. In *Machine Learning: ECML-2001*, number 2167 in Lecture Notes in Artificial Intelligence, pages 13–24. Springer-Verlag, 2001.

[3] P. Brazdil and L. Torgo. Knowledge acquisition via knowledge integration. In *Current Trends in AI, B. Wielinga et al.(eds.), IOS Press, Amsterdam.*, 1990.

[4] W. Davies and P. Edwards. The communication of inductive inferences. In G. Weiß, editor, *Distributed Artificial Intelligence Meets Machine Learning: Learning in Multi-Agent Environments*, volume 1221, pages 223–241. Springer-Verlag, 1997.

[5] L. K. Hansen and P. Salomon. Neural network ensembles. *IEEE Transactions on Pattern analysis and machine intelligence*, 12:993–1001, 1990.

[6] M. Huhns and G. Weiss. About multi-agent learning. *Machine Learning*, 33:1–3, 1998.

[7] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

[8] T. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[9] S. Onta nón and E. Plaza. Learning when to collaborate among learning agents. In L. De raedt and P. Flach, editors, *Machine Learning: ECML-2001. Lecture Notes in artificial Intelligence*, volume 2167, pages 394 – 405, 2001.

[10] S. Ontañón and E. Plaza. A bartering approach to improve multiagent learning. In ACM Press, editor, *Procs of the First Int. Joint Conf. on Autonomous Agents and Muliagent Systems (AAMAS-2002)*, pages 386 – 393, 2002.

[11] E. Plaza and S. Ontañón. Ensemble case-based reasoning: Colaboration policies for multiagent cooperative cbr. In I. Watson and Q. Yang, editors, *CBR Research and Development: ICCBR-2001*, volume 2080, pages 437 – 451, 2001.

[12] E. Plaza and S. Ontañón. Justification-based multiagent learning. In *Proceedings of ICML-2003*, 2003.

[13] A. Prodromidis, P. Chan, and S. Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. In *Book on Advances of Distributed Data Mining, editors Hillol Kargupta and Philip Chan, AAAI press, 2000.*, 2000.

[14] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.