Market-Inspired Approach to Collaborative Learning

Jan Tožička, Michal Jakob, and Michal Pěchouček

Gerstner Laboratory
Department of Cybernetics, Czech Technical University
Technická 2, Prague, 166 27, Czech Republic
{tozicka, jakob, pechouc}@labe.felk.cvut.cz

Abstract. The paper describes a decentralized peer-to-peer multi-agent learning method based on inductive logic programming and knowledge trading. The method uses first-order logic for model representation. This enables flexible sharing of learned knowledge at different levels of abstraction as well as seamless integration of models created by other agents. A market-inspired mechanism involving knowledge trading is used for inter-agent coordination. This allows for decentralized coordination of learning activity without the need for a central control element. In addition, agents can participate in collaborative learning while pursuing their individual goals and maintaining full control over the disclosure of their private information. Several different types of agents differing in the level and form of knowledge exchange are considered. The mechanism is evaluated using a set of performance criteria on several scenarios in a realistic logistic domain extended with adversary behavior. The results show that using the proposed method agents can collaboratively learn properties of their environment, and consequently significantly improve their operation.

1 Introduction

There are two different perspectives from which learning in multi-agent systems can be viewed. The first, perhaps the more pragmatic one views multi-agent systems as a possible tool for solving complex learning problems. The second perspective is driven by the understanding that adaptivity is one of the fundamental properties of any intelligent system. This perspective then considers machine learning as a set of techniques using which intelligent agents and multi-agent systems can adapt in changing environments. Although quite different at the first sight, each emphasizing different priorities and seemingly different goals, both perspectives should ultimately lead to similar core principles and techniques, and should be therefore viewed as dual rather than conflicting.

This is exactly the case with the collaborative learning mechanism described in this article. Although primarily designed to equip multi-agent systems with adaptation abilities, it could be also used as a distributed machine learning algorithm. The algorithm combines first-order logic as a basis for knowledge representation

M. Klusch, M. Rovatsos, and T. Payne (Eds.): CIA 2006, LNAI 4149, pp. 213–227, 2006.

and learning with market principles for decentralized inter-agent coordination. Altogether, the method addresses some of the difficult challenges posed by learning in the multi-agent environment, including the distribution of learning sub-tasks, inter-agent communication, and the coordination of the learning activity.

The proposed method has been implemented within the cognitive-reflective agent framework [1]. In this framework, learned models are represented as encapsulated components that can be integrated with agent's reasoning layer in a plug-and-play manner. This offers several important advantages. The learned theory can be immediately operationalized in guiding agent's behavior. Second, the theory or its parts can be easily exchanged between agents. Finally, agents can rapidly reconfigure their reasoning to match the characteristics of the changing environment.

1.1 Structure of the Paper

In Section 2, we review some of the existing work relevant to our multi-agent learning method. Section 3 then introduces inductive logic programming, which forms the basis of our approach. The proposed collaborative multi-agent learning method based on knowledge trading is described in Section 4. Section 5 evaluates the method on a set of scenarios and discusses the obtained experimental results. Finally, we summarize our work in Section 6.

2 Survey of Existing Work

In this section, we give a brief overview of existing work on multi-agent machine learning and its application to distributed data mining. As our method builds strongly on logic- and market-based approaches, we examine both of them in greater detail.

2.1 Machine Learning in Multi-Agents Systems

The majority of research on learning in multi-agent systems focuses on reactive reward-based approaches and their application to inter-agent coordination [2]. Considerably less work exists on higher-level concept learning and the role of explicit inter-agent communication in multi-agent learning. Panait and Luke [3] present an exhaustive review of cooperative methods for multi-agent learning. They discuss the role of communication in learning, distinguishing between direct and indirect communication.

Weiss [4] proposes a classification scheme distinguishing between three classes of multi-agent learning mechanisms, depending on the amount of cooperation among agents: *multiplication*, *division* and *interaction*. In Weiss'es classification, the collaborative method presented in this papers uses the interaction mechanism. In contrast to the division mechanism which only allows the exchange of raw training examples (see also coactive learning [5]), interaction mechanisms involve higher-level exchange of models created during learning. Apart from

potentially speeding up the learning process, this has an additional significant benefit of protecting the privacy of individual agents, and is therefore crucial in domains where agents have private data sources.

2.2 Logic-Based Learning in Multi-Agent Systems

Kazakov [6] discusses the application of ILP for single-agent learning in the multi-agent setting. Agents individually learn the properties of the environment using a Progol ILP system. In contrast to our approach, no communication between agents takes place.

Hernandez [7] discusses the application of the first-order decision tree induction system ACE to learn about the applicability of plans in the BDI architecture. There is no inter-agent communication beyond plain observation exchange and the learning system is not integrated into agent's reasoning architecture.

Alonso [8] advocates the application of ILP and other logic-based techniques for learning in complex multi-agent domains such as conflict simulations.

2.3 Market-Based Approaches to Learning

Market-based techniques have become a popular approach to coordination in multi-agent and multi-robot systems. However, there is currently only a very limited work on the application of such techniques to multi-agent learning. A notable exception is the work by Wei et al. [9] presenting a market mechanism for the aggregation of the output of multiple learning agents.

2.4 Distributed Data Mining

Distributed data mining is an important application area for the proposed method and multi-agent learning techniques in general. *Data mining* [10] is concerned with the analysis of possibly massive amounts of data. Various distributed data mining methods [11,12,13] have been proposed to address the scalability issues limiting the applicability of centralized data mining techniques. Klusch et al. [14] analyze the benefits of the multi-agent approach to distributed data mining, especially in open, heterogeneous environments with plurality of different data sources and data mining methods.

3 Inductive Logic Programming

In this section, we introduce inductive logic programming which forms the basis of our multi-agent learning method. Inductive Logic Programming (ILP) [15] systems fall into the category of machine learning algorithms. They use domain-specific background information, encoded by means of a predicate logic theory, and pre-classified sample data in the form of first-order ground facts, to construct a predicate logic theory for deriving data classification. In most cases, the first-order logic language for expressing both the background theory and the learned

theories is constrained to a list of Horn clauses, i.e., to the grammar of Prolog programs. In our implementation, the quality of created theories is evaluated using F_1 measure [16].

ILP is a suitable learning method for the use in multi-agent systems (e.g. [6]) because it represents both the input to the learing process, i.e., the examples and the background knowledge, as well the output learned theory in first-order logic (or in its Horn fragment). Due to the expressivity and the well-defined semantics of first-order representations, learned models can be easily exchanged among agents and reused in agent's further learning or reasoning processes. This feature of ILP is particularly important for collaborative multi-agent learning.

In our case, the sets of positive and negative examples are agents' observations classifying situations occurring in agent's environment. The objective of learning is to create a model which can predict these classifications. Background theory B contains agent's common knowledge including the knowledge describing the context (i.e., relational and temporal properties) of training examples. In addition, individual Horn rules in the created theory are assigned weights specifying how many positive and negative observations they cover. These weights are then used during situation classification to get finer than just binary classification.

4 Collaborative Learning with Knowledge Trading

In this section, we describe our collaborative learning method based on knowledge trading. On the single-agent level, the mechanism uses ILP as an inductive method for generating predictive models from examples (observations) an agent receives. On the inter-agent level, the ILP is complemented with a trading mechanism through which agents can trade observations and (sub-)theories. Agents in the system differ with respect to whether they create models and whether and how they trade them with others.

4.1 Knowledge Trading Protocol

Interaction between agents in the system is governed by a variant of the Contract-Net-Protocol (Figure 1). The seller, i.e., the agent offering its theory, sends Call-for-Proposal messages containing the meta-description of its theory to other agents. The meta-description contains the information used to calculate the F_1 measure, i.e., the number of covered positive and negative examples, and the number of all positive examples in the training set.

Each recipient, i.e., a potential buyer evaluates the offered theory and, if it finds the theory useful, replies with a meta-description of the knowledge it wants to offer in exchange. The knowledge offered can be either a set of recipient's observations (i.e., a training set and a background knowledge) or its own generalized theory. Note that agents are not allowed to resell knowledge acquired from other agents.

Next, the seller evaluates received proposals and selects the sellers that have proposed attractive knowledge. It then sends them the *accept* message with its theory and receives the proposed knowledge as a reply.

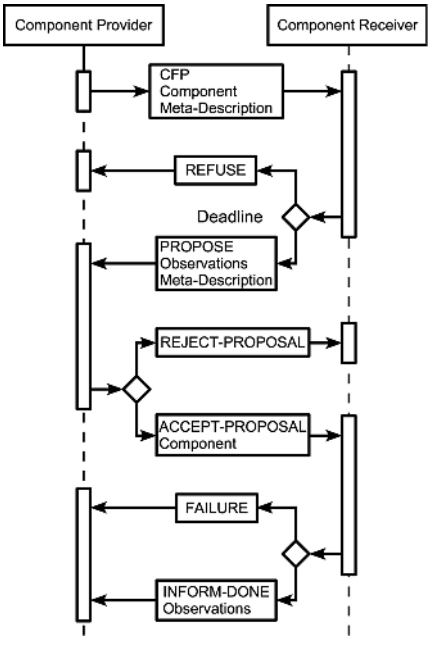

Fig. 1. The knowledge trading protocol used in the collective learning

4.2 Agent Classification

A range of collaborative learning agents can be implemented using the above trading protocol. In order to categorize them, we introduce several agent classification properties:

learning: an agent is called *learning* if it creates its own theory; otherwise it is called *non-learning*

offering: learning agents can be further classified as *offering* or *non-offering* depending on whether or not they actively offer the learned model (i.e., whether they can play the role of the provider in the trading protocol)

trading: an agent is called *trading* if it participates in knowledge barter when asked by an offering agent; otherwise it is called *non-trading*; furthermore, an agent can either trade *rules* or raw *observations*

Table 1 summarizes possible types of agents under the proposed classification schema.

4.3 Types of Agents

Not all possible types of agents are used in our collaborative learning mechanism. A^D, for example, is a totally non-adaptive agent, and is therefore not an interesting member of a collaboratively learning agent society. Similarly, agent

Table 1. Types of agents that can be implemented based on the three introduced classification properties: *learning*, *offering* and *trading*

Types of Agents		Non-Trading	Trading	
			Observations	Rules
Non-Learning		A^D	A^L	N/A
Learning and	Non-Offering	A^S	A^{O-}	A^{R-}
	Offering	A^P	A^O	A^R

A^P never participates as a recipient in a knowledge barter. Although its behavior might be interesting in some heterogeneous societies, it is currently not used in our system. Finally, as preliminary tests with the agents of type A^{O-} and A^{R-} have not yield good results, the agents have been excluded from experiments discussed in this paper.

The remaining agent types (A^S , A^O , A^R , and A^L) have been experimentally evaluated and are described in greater detail below. Experimental results involving these agents are then provided in Section 5.

A^S – Simple Agent. does not communicate with other agents and therefore does not participate in the collaborative learning process. The agent learns in isolation, creating its own theory based solely on its own observations. It is therefore expected to converge more slowly to a target theory than collaborative agents. It has been implemented for comparison purposes only.

A^O – Observation-Based Agent. offers its created theory for trading and buys theories offered by other agents in exchange for its observations. It buys theories of all sizes (i.e., even theories covering only a small number of positive examples), however, in exchange it offers only a subset of its observations covering the same number of positive examples as the theory bought does. The agent thus uses the number of covered positive examples as the measure of theory quality. This is an appropriate choice in the domain considered (see Section 5.2) as observations representing positive examples contain the most valuable knowledge. In other domains, a different theory quality measure can be more suitable.

A^R – Rule-Based Agent. trades theories for theories, both as the offering agent or as the recipient in a knowledge barter. In both cases, it accepts all theories that are better than its own. Theories worse than its own theory are accepted only with the probability equal to the ratio between the quality of the offered theory and the quality of agent's own theory (which is always lower than 1).

Bought theories are appended to agent's own theory, and the positive examples covered by the newly acquired theory are removed from agent's observation set. The ILP learning algorithm is subsequently invoked to derive a theory covering only the remaining, uncovered observations. This significantly speeds up the

learning process because the time complexity of ILP grows exponentially with the size of the training set.

Communication bandwidth required by rule-based agents is significantly lower than the bandwidth needed by observation-based agents. This is due to *compression* performed by the inductive learning algorithm: in most cases, the size of the generalized model is significantly smaller than the size of the training set from which it was generated.

A^L – Lazy Agent. does not create a theory on its own. Instead, similarly to A^O agents, it buys other agents' theories in exchange for its observations. The lazy agent is very lightweight regarding its computational requirements. However, it needs higher communication bandwidth to communicate its observations.

5 Experiments

This section describes the empirical tests we have conducted to evaluate the performance of the proposed collaborative learning mechanism. First, we briefly outline the implementation of the method and introduce the domain in which learning takes place. Next, we describe experiments performed involving different types of agents and collaborative communities. Finally, we present and discuss the experimental results obtained.

5.1 Reflective-Cognitive Agent Architecture

The proposed learning method has been implemented within the reflective-cognitive agent architecture, a modular Java-based architecture for the design and implementation of autonomous intelligent agents [1]. The reflective-cognitive agent is composed of two parts (Figure 2):

- **the reasoning layer** implements agent decision-making. The layer is implemented using a modular approach based on the component architecture. Agent's reasoning can be reconfigured by adding/removing new reasoning components in run-time. ILP model is an example of the reasoning module that can be integrated with the agent reasoning cycle.
- **the reflective-cognitive (RC) layer** manages the reasoning layer. It implements closed-loop adaptation by monitoring agent's performance and modifying the reasoning layer, in order to optimize agent's operation in the changing environment. The modification is of agent's behavior is achieved through adding, removing and possibly fine-tuning components of the reasoning layer. The RC layer can also communicate with other agents' RC layers in order to exchange and integrate components created by other RC agents in the system.

5.2 Domain Description

The domain ACROSS [17] used in the empirical evaluation is a logistic scenario extended with adversarial behavior. In the domain, truck transporter agents carry

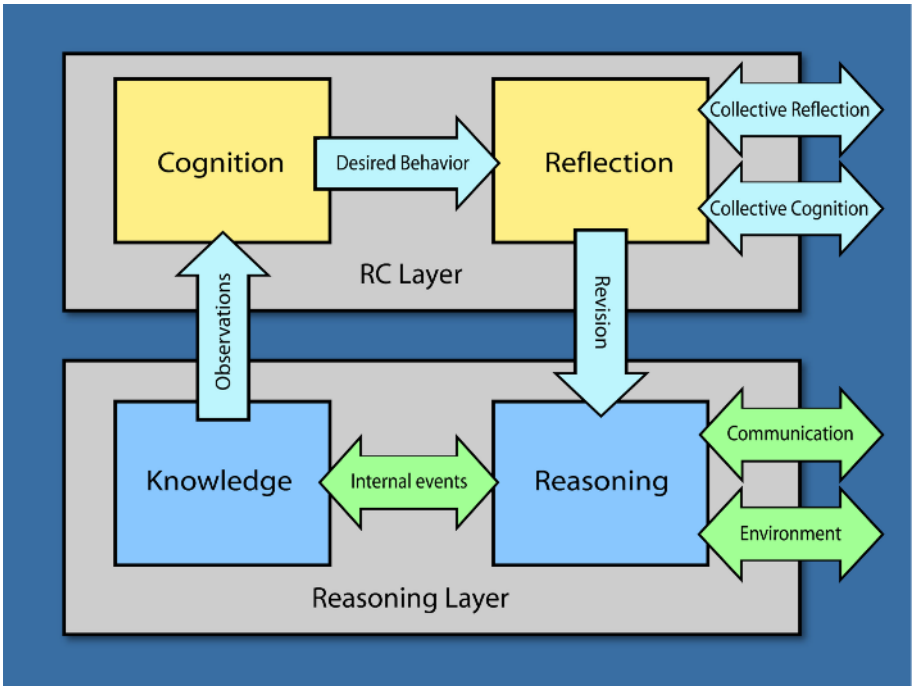

Fig. 2. The scheme of Reflective-Cognitive agent's architecture

goods between producers and consumers. The transporter agents are able to form coalitions in order to improve their chances when competing for transport tasks.

In addition to the transporter agents, adversarial *bandit agents* that can attack and rob transporter agents are present in the domain. The activity of bandit agents is not the same everywhere. Instead, each bandit agent has a set of preferences specifying in which areas and under which conditions it attacks. These preferences are described by a relational theory taking into account the properties of the road network in the scenario. Bandit agents also have some restrictions on the transporter agents and the situations in which they attack (e.g., *transporters of tribe Northlanders carrying cargo to a location not producing this cargo*). The situation description is the part of the agent's observations that do not belong to the training set used for ILP learning. A training example is generated whenever an agent is robbed (a positive example represented by predicate *holdup*), or when it safely passes a road (a negative example represented by predicate *noholdup*).

In experiments, transporter agents try to learn bandits' restrictions in order to operate more safely. Each transporter agent is provided with an ILP system, using which it generates a theory predicting bandits' behavior. It does not attempt to create a theory covering all possible circumstances but only those relevant to its properties and regions in which it operates (e.g., transporter's tribe or its home city's region).

5.3 Example

Let us illustrate learning in ACROSS with an example. In this case, a bandit agent uses the following rule do decide whether or not to attack:

```
attack(Transporter):-
    endCity(Transporter, C1),
    cityRegion(C1, 'Central'),
    startCity(Transporter, C2),
    cityRegion(C2, 'Central'),
    notEqual(C1, C2).
```

This bandit agent attacks only transporter agents carrying goods between two different cities in the `Central` region.

Operating in this domain, a transporter agent could learn the following rule[1] representing its view of bandit agent's behavior:

```
attack(Transporter):-
    endCity(Transporter, C1),
    cityTribe(C1, 'Midlanders'),
    startCity(Transporter, C2),
    cityPopulation(C2, 'village').
```

On the first sight, the rule learned by the transporter agent looks quite different to the actual rule guiding the bandit agent's behavior. However, because of the fact that most locations in the `Central` region belong to the `Midlanders` tribe (and vice versa), and some locations next to the border of the `Central` region are villages, this rule in fact closely approximates the actual behavior of the bandit agent.

Note that the rule learned by the agent uses variables and a conjunction of different predicates to concisely express a condition that covers a large number of specific situations. The same condition would have to be represented as a long enumeration of specific cases if relational, logic-based learning was not used.

5.4 Experiment Scenario Setup

We have used simple A^S , observation-based A^O , rule-based A^R and lazy A^Lagents in our experiments (see Section 4.3 for the detailed description of agent types). Using these agents, we have designed two sets of scenarios with (i) homogeneous, and (ii) heterogeneous societies of agents.

Homogeneous society consists of N agents of the identical type. Only learning agents are considered for homogenous societies as societies consisting of non-learning agents only would have no adaptation ability, and are thus not interesting for our study. Altogether, we thus have the following three agent societies:

[1] This is just an example, the learned model usually consists of several rules of this kind.

- **SC-1** consists of N simple agents
- **SC-2** consists of N observation-based agents
- **SC-3** consists of N rule-based agents

Note that while observation-based agents solely share their observations, rule-based agents share created models, and use them to filter out covered positive examples from their training sets. As a result, each rule-based agent tries to cover a different area of the whole learning space. This leads to the emergence of specialization in the agents, and to a spontaneous decomposition of the learning task based solely on the decentralized knowledge trading mechanism. In all these scenarios, we have evaluated average properties over all N participating agents.

Heterogeneous societies have been used in the second set of experiments. Out of a number of possible combination, we have decided to evaluate societies consisting of a mixture of observation-based and lazy agents. This decision was motivated by the need to evaluate the performance trade-offs of lazy agents. Specifically, we have considered the following two societies:

- **SC-4** consists of a single lazy agent and observation-based agents as the rest, i.e., $(N-1) \times A^O + 1 \times A^L$
- **SC-5** consists of a balanced mixture of lazy and observation-based agents, i.e., $(N/2) \times A^O + (N/2) \times A^L$

In the both experimental scenarios we have focused on the behavior of lazy agents.

5.5 Evaluation Criteria

We have measured the following properties:

prediction quality is defined as the number of robberies. This measure shows the number of agent's false negative predictions, i.e., how many times a road classified as *safe* was not successfully passed.

communication load is measured as the amount of data transferred during knowledge exchange.

computational load is measured as the amount of CPU time consumed by the ILP system. This property generally depends on two factors: (i) the number of ILP invocations, and (ii) the length of each ILP run.

5.6 Results

Let us now present experimental results obtained on the described scenarios using the above defined evaluation criteria. All results are summed over tens of simulation cycles and averaged over five simulation runs.

Results for the Scenarios with Homogeneous Society of Agents. In the case of scenarios involving homogenous societies, we have measured the average per-agent value of each evaluation criteria. Each society consisted of five

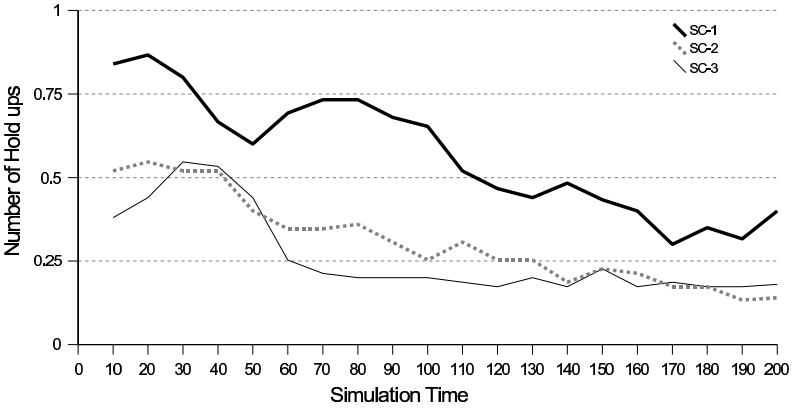

Fig. 3. Number of robberies during the simulation. This illustrates how well the learned theory covers positive examples, i.e., dangerous roads in our scenario. The average number of robberies for A^D agent without learning capability is approximately 0.8.

learning transporter agents and three bandit agents randomly passing the map and robbing transporters they met whenever their restrictions allowed it.

Graph 3 shows how fast the agents adapt to the domain in the sense of minimizing the number of robberies. We can see that all agents improve their behavior (agents without learning capability have an expected robbery probability of 0.8 approximately), but the agents sharing their knowledge learn much faster, particularly at the beginning of the simulation – in as little as ten cycles the number of robberies was decreased to nearly one half. Both these observations were expected, unlike the rather surprising one that A^O agents only slightly outperformed A^R agent during the first half of the experiment. At the end of the experiment agents perform similarly well.

Graph 4 shows how many bytes were sent on average by each agent communicating its knowledge. A^S agents (in SC-1 scenario) do not communicate at all. A^O agents (in SC-2 scenario) communicate approximately 20-times more on average then A^R agents (in SC-3 scenario). While during the initial 50 cycles this ratio is less then 10:1 (A^O:A^R), it rises up to approximately 40:1 in the middle of the experiment, and finally converges to 20:1.

Finally, Graph 5 shows the computational demand of ILP learning. A theory induction operation is started after each holdup event to ensure that the agent would not repeat its misjudgment. A^S agents (in SC-1 scenario) consume a lot of resources because they run time-consuming ILP even if their knowledge is only slightly improved – this can be improved using batch learning where a new theory would be created only if the agent has acquired at least some minimal number of new observations, on the expense of a possible increase in the number of robberies. A^O (in SC-2 scenario) and A^R (in SC-3 scenario) agents have similar computational requirements on average, though in the first half of experiments A^R agents are less time-consuming. This is caused by two factors: first, the

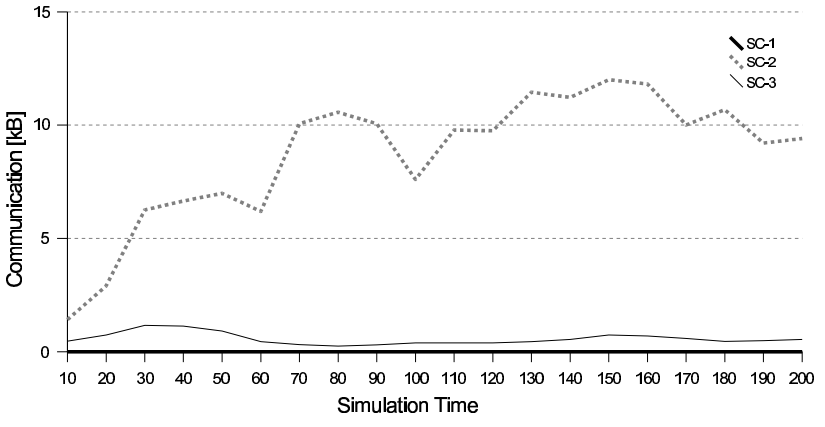

Fig. 4. Communication traffic during collective reasoning. It is the number of kilobytes of contents of messages during knowledge exchange. Note that there is no communication in SC-1 scenario.

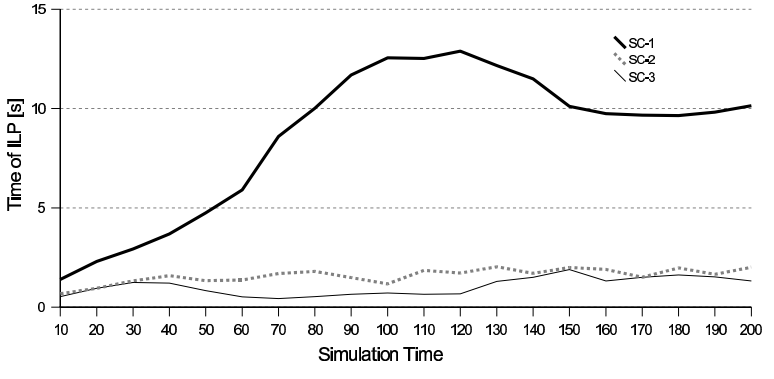

Fig. 5. Time needed to create theories using ILP system on state-of-the-art machine

training set of an A^O agent grows much faster as it receives other observations[2]; second, the filtration of positive examples used by A^R agents very often filters the positive examples out of the trainig set. A society of A^R agents can be therefore recommended to run on slower machines: even if the ILP ran more often (when aggregated over all the agents), the time-consumption of individual runs was smaller than in the case of A^O agents.

Results of Scenarios with the Heterogeneous Agent Society. A^L agents in SC-4 scenario always buy new rules offered by other agents in the community (A^O in our case). The better the rules are, the higher number of positive examples they cover, and therefore the more expensive they are. As a result, A^L agents

[2] Note that this can lead to flooding in some cases, e.g., when these observations are irrelevant for the receiving agent.

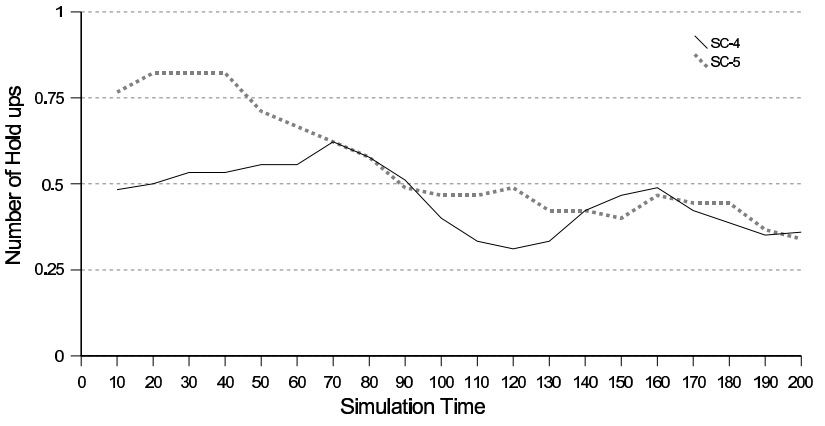

Fig. 6. Number of hold-ups during the simulation in SC-4 and SC-5 scenarios. It illustrates how well the created theory covers positive examples, or dangerous roads in our scenario.

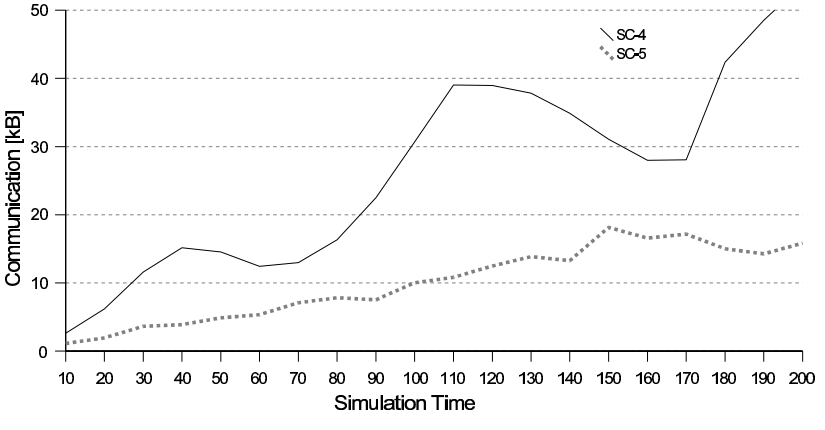

Fig. 7. Communication traffic during collective reasoning in SC-4 and SC-5 scenarios. It is the number of kilobytes of contents of messages during knowledge exchange.

have to send more observations in exchange. This leads to an unlimited growth in communication traffic until all supplying agents have perfect theories and do not improve them any more.

The last graph (Figure 6) demonstrates that if there is only a small number of A^L agents in the community (SC-4 scenario), they are fairly successful from the beginning of the simulation but later they improve very slowly. A higher proportion of A^L agents in the community (SC-5 scenario) causes slower learning in the beginning, though later it reaches the performance of SC-4 scenario. This is partially caused by the difficulty of the learning task because $N/2$ agents were able to cover dangerous roads with good accuracy.

Graph 7 illustrates the growth of communication traffic during the simulation. Higher communication load in SC-4 scenario corresponds to a higher number of A^O agents offering their knowledge to A^L agents. Note that the time needed to run ILP in SC-4 and SC-5 scenarios is zero as we measured A^L (lazy) agents only.

6 Conclusions

In this paper, we have presented collaborative learning agents that can share their knowledge using a simple trading protocol. Depending on their roles in the trading protocol, we have identified several types of knowledge trading agents. We have evaluated the performance of both homogenous and heterogeneous communities of such agents with respect to several criteria, including the quality of learned models and the communication and computational resources required. The experiments have shown that agents trading generalized models outperform agents exchanging raw observations only. Even the latter, however, outperform non-collaborative agents in terms of model quality and computational resources required. Altogether, the proposed mechanism allows effective distributing learning without the need for a central coordinator or other centralized resources. In consequence, it enables the creation of robust and scalable peer-to-peer learning systems.

Acknowledgement

We gratefully acknowledge the support of the presented research by Army Research Laboratory project N62558-03-0819.

References

1. Foltýn, L., Tožička, J., Rollo, M., Pěchouček, M., Jisl, P.: Reflective-cognitive architecture: From abstract concept to self-adapting agent. In: DIS '06: Proceedings of the Workshop on Distributed Intelligent Systems, IEEE Comp. Soc. (2006)
2. Kudenko, D., Kazakov, D., Alonso, E., eds.: Adaptive Agents and Multi-Agent Systems II: Adaptation and Multi-Agent Learning. In Kudenko, D., Kazakov, D., Alonso, E., eds.: Adaptive Agents and Multi-Agent Systems II: Adaptation and Multi-Agent Learning. Volume 3394 of LNCS., Springer (2005)
3. Panait, L., Luke, S.: Cooperative multi-agent learning: The state of the art. Autonomous Agents and Multi-Agent Systems **11**(3) (2005) 387–434
4. Weiss, G., Dillenbourg, P.: What is 'multi' in multi-agent learning? In: P. Dillenbourg (Ed) Collaborative-learning: Cognitive and Computational Approaches. Elsevier, Oxford (1999) 64–80
5. Grecu, D.L., Becker, L.A.: Coactive Learning for Distributed Data Mining. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York, NY (1998) 209–213
6. Kazakov, D., Kudenko, D.: Machine learning and inductive logic programming for multi-agent systems. In: Multi-Agent Systems and Applications. Volume 2086 of LNAI., Prague, Czech Republic, Springer Verlag (2001) 246–270

7. Guerra-Hernandez, A., Fallah-Seghrouchni, A., Soldano, H.: Learning in BDI multi-agent systems. In: in Proceedings of CLIMA 2003. (2004) 185–200
8. Alonso, E., d'Inverno, M., Kudenko, D., Luck, M., Noble, J.: Learning in multi-agent systems. Knowledge Engineering Review **16**(3) (2001) 277–284
9. Wei, Y.Z., Moreau, L., Jennings, N.R.: Recommender systems: a market-based design. In: AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, New York, NY, USA, ACM Press (2003) 600–607
10. Hand, D., Mannila, H., Smyth, P.: Principles of Data Mining. A Bradford Book The MIT Press Cambridge (2001)
11. Kargupta, H., Chan, P., eds.: Advances in Distributed and Parallel Knowledge Discovery. In Kargupta, H., Chan, P., eds.: Advances in Distributed and Parallel Knowledge Discovery, MIT/AAAI Press (2000)
12. Park, B., Kargupta, H.: Distributed Data Mining: Algorithms, Systems, and Applications. In Ye, N., ed.: Data Mining Handbook. IEA (2002) 341–358
13. Giannella, C., Bhargava, R., Kargupta, H.: Multi-agent systems and distributed data mining. In Klusch, M., Ossowski, S., Kashyap, V., Unland, R., Laamanen, H., eds.: Cooperative Information Agent VIII. LNAI 3191, Springer-Verlag, Heidelberg (2004) 1–15
14. Klusch, M., Lodi, S., Moro, G.: Agent-based distributed data mining: The kdec scheme. In: AgentLink. Number 2586 in LNCS, Springer (2003)
15. Muggleton, S., Raedt, L.D.: Inductive logic programming: Theory and methods. Journal of Logic Programming **19/20** (1994) 629–679
16. van Rijsbergen, C.J.: Information Retrieval. Butterworths, London (1979)
17. Šišlák, D., Rehák, M., Pěchouček, M., Rollo, M., Pavlíček, D.: $\mathcal{A}$-**globe**: Agent development platform with inaccessibility and mobility support. In Unland, R., Klusch, M., Calisti, M., eds.: Software Agent-Based Applications, Platforms and Development Kits, Berlin, Birkhauser Verlag (2005) 21–46