Cooperative Learning Using Advice Exchange

Luís Nunes[1,2], Eugénio Oliveira[1]

[1] Laboratório de Inteligência Artificial e Ciência de Computadores (LIACC) – Núcleo de Inteligência Artificial Distribuída e Robótica (NIAD&R), Faculdade de Engenharia da Universidade do Porto (FEUP),Av. Dr. Roberto Frias 4200-465, Porto, Portugal.
[2] Instituto Superior de Ciências do Trabalho e da Empresa (ISCTE), Av. Forças Armadas, Edíficio ISCTE, 1649-026, Lisboa, Portugal
eco@fe.up.pt, Luis.Nunes@iscte.pt

Abstract. One of the main questions concerning learning in a Multi-Agent System's environment is: "(How) can agents benefit from mutual interaction during the learning process?" This paper describes a technique that enables a heterogeneous group of Learning Agents (LAs) to improve its learning performance by exchanging advice. This technique uses supervised learning (backpropagation), where the desired response is not given by the environment but is based on advice given by peers with better performance score. The LAs are facing problems with similar structure, in environments where only reinforcement information is available. Each LA applies a different, well known, learning technique. The problem used for the evaluation of LAs performance is a simplified traffic-control simulation. In this paper the reader can find a summarized description of the traffic simulation and Learning Agents (focused on the advice-exchange mechanism), a discussion of the first results obtained and suggested techniques to overcome the problems that have been observed.

1 Introduction

The objective of this work is to contribute to give a credible answer to the following question: "(How) can agents benefit from mutual interaction during the learning process, in order to achieve better individual and overall system performances?".

The objects of study are the interactions between the Learning Agents (hereafter referred as agents for the sake of simplicity) and the effects these interactions have on individual and global learning processes. Interactions that affect the learning process can take several forms, in Multi-Agent Systems (MAS). These different forms of interaction can range from the indirect effects of other agents' actions (whether they are cooperative or competitive), to direct communication of complex knowledge structures, as well as cooperative negotiation of a search policy or solution.

The most promising way in which cooperative learning agents can benefit from interaction seems to be by exchanging (or sharing) information regarding the learning

process itself. As observed by Tan [1] agents can exchange information regarding several aspects of the learning process: a) the state of the environment, b) episodes (state, action, reward triplets), or c) internal parameters and policies.

Exchanging environment states may be interpreted as if each agent has extra sets of sensors spread out in the environment, being able to have a more complete view of the external state. This larger view of the state-space may require either pre-acquired knowledge on how to interpret this information and integrate it with its own view of the environment's state, or simply be considered as extra input providing a wider range of information about the state. Techniques that use this strategy may be adequate to solve the problem of partially observable states in environments where this situation creates serious problems to learning or cooperation amongst agents.

Episode exchange requires that the agents are (or have been) facing similar problems, requiring similar solutions and may lead to large amounts of communication if there are no criteria regulating the exchange of information. In the limit case, where all agents share all the episodes, this process can also be seen as a single learning system, and produce very little new knowledge. In fact, the exchange of too much data may lead all the agents to follow the same path through the search space, wasting valuable exploration resources. Nevertheless, the exchange of information has proved to be beneficial if used with care, as shall be demonstrated.

Sharing internal parameters requires that agents have similar internal structures, so that they can easily map their peers' internal parameters into their own, or that they share a complex domain ontology. If there are no restrictions to communication, and the user can be sure that a particular learning algorithm is more suitable than others to solve the problem at hand, or if there is a reliable way of mapping the knowledge acquired by one agent to its peers, this type of information exchange can be very effective.

The question of exchanging information during learning is not only: "what type of information to exchange?" but also "when to exchange information?", "how much information is it convenient to exchange?", "how to use shared information?" and "what is the reliability of each piece of information?".

When considering human cooperative learning in a team, a common method to improve one's skills is to ask for advice at critical times, or to request a demonstration of a solution to a particular problem to someone who is reputed to have better skills in the subject. During this process several situations may occur:

- The advisee evaluates the capacity of the elements of a group of potential advisors to provide advice on a specific situation, then selects an advisor and explains the problem.
- The advisor seeks a solution, selects which information to give and tries to present it on a format that is understandable by the advisee. The advisor can also give meta-information regarding the validity and limitations of that advice.
- The advisee pre-evaluates the advice based on his past experience and in the trust he has in the advisor, interprets the advice and applies it to its own situation. Then he evaluates the results, and finally updates its opinion concerning the advisors' ability to help in a given situation.

This process, (or selected parts of it), is what we are attempting to translate into the realm of Multi-Agent Systems Learning (MASL).

Another important point in human learning is that different people specialize on different things, either because of their differing talents or as a consequence of their working experience that has forced them to work more in certain problems. This means that some people will be more efficient at dealing with some specific problems than others. In MAS, especially when considering agents with different learning capabilities, this is another point that can be explored. It is common that some kind of specialization occurs, either because a particular learning technique is more fit to solve a certain problem or simply because the dynamics of the environment have caused one agent to have more experience than others in certain areas of the problem.

The initial work on MASL, reported here, is mainly concerned with the effect of exchanging advice in a heterogeneous group of agents, where each one is dealing with problems with similar structure, but in which the actions of one agent do not have an impact on the state sensed by other agents. This scenario can be found in several application domains in which agents need to solve similar problems but do not need to share resources to solve them. One example might be the internet web-crawlers, that can learn from the experience of their peers but, apart from the information exchanged amongst them, have little impact on the state as seen by their peers. In this paper's future work section several extensions to this problem are previewed.

The main difference from other related work is in the use of agents with different learning strategies, as opposed to the study of cooperative Q-Learners that is the most common approach in the related work. The authors believe that the heterogeneity of the group can help to overcome the problems of the "No Free Lunch Theorem" and provide better response to difficult distributed control tasks in which learning can be advantageous.

In the experiments, agents selectively share episodes by requesting advice for given situations to other agents whose score is, currently, better than their own in solving a particular problem. The advisors are not pre-trained experts. All agents are started at the same time and run synchronously.

Considering the several possibilities for exchanging information regarding the learning process, discussed in the previous section, this option seemed the most promising for the following reasons: a) Sharing of episodes does not put heavy restrictions on the heterogeneity of the underlying learning algorithms and may be achieved using a simple communication mechanism; b) Having different algorithms solving similar problems may lead to different forms of exploration of the same search space, thus increasing the probability of finding a good solution; c) It is more informative and less dependent on pre-coded knowledge than the exchange of environment's states.

Experiments were conducted with a group of agents embedded in a simplified simulation of a traffic-control problem to test the advantages and problems of advice-exchange during learning. Agents are heterogeneous (i.e., each applies a different learning mechanism, unknown to others). Each individual agent uses a standard version of a well know, sub-symbolic, learning algorithm (so far the set of algorithms used is restricted to: Random Walk, Evolutionary Algorithms, Simulated Annealing, and Q-Learning).

The heterogeneous nature of the group makes communication of internal parameters or policies difficult to use since sharing this information would require agents to translate their internal knowledge to a common format. Despite the fact that this is an interesting question it is beyond the scope of the current research.

The exchanged information is: current state (as seen by the advisee); best response that can be provided to that state (by the advisor agent); current and best scores, broadcasted at the end of each training stage (epoch).

The problem chosen to test the use of advice-exchange has, as most problems studied in MASL, the following characteristics: a) Analytical computation of the optimal actions is intractable; b) The only information available to evaluate learning is a measure of the quality of the present state of the system; c) The information regarding the quality of the state is composed of a local and a global component; d) The same action executed by a given agent may have different consequences at different times, even if the system is (as far as the agent is allowed to know) in the same state; e) The agent has only a partial view of the problem's state.

The simplified traffic-control problem chosen for these experiments requires that each agent learn to control the traffic-lights in one intersection under variable traffic conditions. Each intersection has four incoming, and four outgoing, lanes. One agent controls the four traffic lights necessary to discipline traffic in one intersection. In the experiments reported here, the crossings controlled by each of the agents are not connected to each other.

The learning parameters of each agent are adapted using two different methods: a reinforcement-based algorithm, using a quality measure that is directly supplied by the environment, and supervised learning using the advice given by peers as the desired response. Notice that the term "reinforcement-based" is used to mean: "based on a scalar quality/utility feedback", as opposed to supervised learning which requires a desired response as feedback. The common usage of the term "reinforcement learning", that refers to variations of temporal difference methods [2], is a subclass of reinforcement-based algorithms, as are, for instance, most flavours of Evolutionary Algorithms.

In section 2 the reader can find references to related work. Section 3 contains a brief description of the experimental setup, focused on the advice-exchange algorithm. Section 4 concerns the discussion of the initial results, and finally in section 5, some conclusions and a preview of the future work to be done in this direction.

2 Related Work

The work on cooperative learning had some important contributions in the early nineties with the results published by Whitehead [3], Lin [4] and Tan [1].

All these works focused on cooperation of Learning Agents that use variations of Q-Learning [5]. Whitehead has experimented two cooperative learning mechanisms: Learning with an External Critic (LEC) and Learning By Watching (LBW). The first, (LEC), is based on the use of an expert automated critic that provides feedback con-

cerning the agent's actions more frequently than the environment would, while the second, (LBW), learns vicariously by watching other agent's behaviour (which is equivalent to sharing state, action, quality triplets). This work proves that the complexity of the search mechanisms of both LEC and LBW is inferior to that of standard Q-Learning for an important class of state-spaces. Experiments reported in [6] support these conclusions.

Lin uses a human teacher to improve the performance of two variants of Q-Learning. This work reports that the "advantages of teaching should become more relevant as the learning task gets more difficult". Results in variants of the maze problem show that teaching does improve learning performance in the harder task, although it seems to have no effect on the performance on the easier task.

Tan addressed the problem of exchanging information during the learning process amongst Q-Learning agents. This work reports the results of sharing several types of information amongst a group of agents in the predator-prey problem. Experiments were conducted in which agents shared policies, episodes (state, action, quality triplets), and sensation (state). Although the experiments use solely Q-Learning in the predator-prey domain, the author believes that: "conclusions can be applied to cooperation among autonomous learning agents in general". Conclusions point out that "a) additional sensation from another agent is beneficial if it can be used efficiently, b) sharing learned policies or episodes among agents speeds up learning at the cost of communication, and c) for joint tasks, agents engaging in partnership can significantly outperform independent agents, although they may learn slowly in the beginning". The results reported by Tan also appear to point to the conclusion that sharing episodes with peers is beneficial and can lead to a performance similar to that obtained by sharing policies. Sharing episodes volunteered by an expert agent leads to the best scores in some of the presented tests, significantly outperforming most of the other strategies in the experiments.

After these first, fundamental, works, several variants of information sharing Q-Learners appeared reporting good results in the mixture of some form of information-sharing and reinforcement learning. Mataric [7] reports on the use of localized communication of sensory data and reward as a way to overcome hidden state and credit assignment problems in groups of Reinforcement Learning agents involved in a cooperative task. The experiments conducted in two robot problems, (block pushing and foraging) show improvements in performance on both cases.

Several researchers investigated the subject of using an expert automated teacher. Baroglio [8] uses an automatic teacher and a technique called "shaping" to teach a Reinforcement Learning algorithm the task of pole balancing. Shaping is defined as a relaxation of the evaluation of goal states in the beginning of training, and a tightening of those conditions in the end. Clouse [9] uses an automatic expert trainer to give a Q-Learning Agent actions to perform, thus reducing the exploration time. Brafman and Tenemholtz [10] use an expert agent to teach a student agent in a version of the "prisoner's dilemma". Both authors use variations of Q-Learning. Price and Boutilier [11] have demonstrated that the use of learning by imitating one (or several) expert agent(s) produces good results, in variants of the maze problem. Again, Q-Learning agents are used. Berenji and Vengerov [12] report analytical and experimental results

concerning the cooperation of Q-Learning agents by sharing quality values amongst them. Experiments were conducted in two abstract problems. Results point out that limitations to cooperative learning described in [3] can be surpassed successfully, under certain circumstances, leading to better results than the theoretical predictions foresaw.

Learning joint actions has also been investigated by several research groups. Claus , Boutilier [13] and Kapetanakis, Kudeneko [14] have worked in this problem using Q-Learning agents that learn from both their actions and their peers'.

Using a human teacher to improve the learning performance of an agent at a given task has also been a topic to which some researchers have devoted their attention. Maclin and Shavlik [15] use human advice, encoded in rules, which are acquired in a programming language that was specifically designed for this purpose. These rules are inserted in a Knowledge Based Neural Network (KBANN) used in Q-Learning to estimate the quality of a given action. Mataric, [16], reports several good results using human teaching and learning by imitation in robot tasks. Experimental results can be found in [17] [18] [19].

In the area of Machine Learning (ML), some interesting experiments have also been conducted that are related to this work. Provost and Hennessy [20] use cooperative learning, partitioning the data amongst a group of Distributed Rule Learners (each performing general-to-specific beam search) to speedup learning for tasks with large amounts of data. Hogg and Williams [30] have experimented in using cooperative search with a mixture of methods (depth-first, backtracking search and heuristic repair) to solve hard graph coloring problems. The learners exchange information on partial solution, and the results report that "even using simple hints they [, the learners,] can improve performance".

Simultaneous uses of Evolutionary Algorithms [21][22] and Backpropagation [23] are relatively common in ML literature, although in most cases Evolutionary Algorithms are used to select the topology or learning parameters, and not to update weights. Some examples can be found in [24] and [25]. There are also reports on the successful use of Evolutionary Algorithms and Backpropagation simultaneously for weight adaptation [26][27][28]. Most of the problems in which a mixture of Evolutionary Algorithms and Backpropagation is used are supervised learning problems, i.e., problems for which the desired response of the system is known in advance (not the case of the problem studied in this paper). Castillo et al. [29] obtained good results in several standard ML problems using Simulated Annealing and Backpropagation, in a similar way to that which is applied in this work. Again, this was used as an add-on to supervised learning to solve a problem for which there is a well known desired response.

The use of learning techniques for the control of traffic-lights can be found in [31] [32] and [33].

3 Experimental Setup

This section will briefly describe the internal details of the traffic simulation, the learning techniques and the advice-exchange algorithm. A more detailed description of the traffic simulation and learning techniques can be found in [34].

3.1 The Traffic Simulator

The traffic simulator environment is composed of lanes, lane-segments, traffic-lights (and the corresponding controlling agents), and cars. The latest are not realistically modeled, having infinite breaking capabilities and being unable to perform any illegal maneuver.

Cars are inserted at the beginning of each lane with a probability that varies in time according to a function with different parameters for each lane. Cars are removed when they reach an extremity of any of the outgoing lane-segments, after having passed through the scenario.

At the beginning of each green-yellow-red cycle, the agents that control each crossing observe the local state of environment and decide on the percentage of green-time (g) to attribute to the North and South lanes, the percentage of time attributed to the East and West lanes is automatically set at $(1 - g)$. Yellow-time is fixed. The environment is represented as real-valued vector. The first four components represent the ratio of the number of incoming vehicles in each lane relative to the total number of incoming vehicles in all lanes. The four remaining values represent the lifetime of the incoming vehicle that is closest to the traffic-light. This state representation is similar to the one that was reported to have produced some of the best results in the experiments conducted by Thorpe [32] for the same type of problem (learning to control traffic-lights at an intersection).

The quality of service of each traffic-light controller is inversely proportional to the average time cars take to cross the traffic light since their creation in the beginning of the lane.

The car generation parameters in the traffic simulator proved difficult to tune. Slight changes led to simulations that were either too difficult (no heuristic nor any learned strategy were able to prevent major traffic jams), or to problems in which both simple heuristics and learned strategies were able to keep a normal traffic flow with very few learning steps.

3.2 Learning Agents

This section contains a summarized description of the learning algorithms used by each of the agents involved in the traffic-lights control experiments, as well as the heuristic used for the fixed strategy agent.

3.2.1 Stand-alone agents

The stand-alone versions of the learning agents are used to provide results with which the performance of advice-exchanging agents could be compared. The stand-alone agents implement four classical learning algorithms: Random Walk (RL), which is a simple hill-climbing algorithm, Simulated Annealing (SA), Evolutionary Algorithms (EA) and Q-Learning (QL). A fifth agent was implemented (HEU) using a fixed heuristic policy. As the objective of these experiments was not to solve this problem in the most efficient way, but to evaluate advice-exchange for problems that have characteristics similar to this, the algorithms were not chosen or fine-tuned to produce the best possible results for traffic-control. The choice of algorithms and their parameters was guided by the goal of comparing the performance of a heterogeneous group of learning agents, using classical learning strategies, in a non-deterministic, non-supervised, partially-observable problem, with and without advice-exchange.

All agents, except QL and HEU, adapt the weights of a small, one hidden layer, neural network.

The Random Walk (RW) algorithm simply disturbs the current values of the weights of the neural network by adding a random value of a magnitude that is decreased throughout the training. At the end of an epoch, the new set of parameters is kept if the average quality of service in the controlled crossing during that epoch is better than the best average quality achieved so far. Simulated Annealing (SA), [35], works in a similar way to Random Walk, but it may accept the new parameters even if the quality has diminished. The probability of acceptance is given by a Boltzman distribution with decaying temperature.

Evolutionary Algorithms (EA), [21], were implemented in a similar way to the one described in [36], which is reported to have been successful in learning to navigate in a difficult variation of the maze problem by updating the weights of a small Recurrent Artificial Neural Network. This implementation relies almost totally in the mutation of the weights, in a way similar to the one used for the disturbance of weights described for RW and SA. Each set of parameters (specimen), which comprises all the weights of a neural network of the appropriate size, is evaluated during one epoch. After the whole population is evaluated, the best n specimens are chosen for mutation and recombination. An elitist strategy is used by keeping the best $b << n$ specimens untouched for the next generation. The remainder of the population is built by mutation and a simple type of recombination, (choosing randomly from each of the parents entire layers of neural network weights).

The Q-Learning (QL) [5] variation that was implemented uses a lookup table (and discrete mappings of the state-space and actions) with an entry for each state-action pair in which the expected utility is updated in the usual way for one level Q-Learning. The values of the learning rate, in the different experiments, were decayed throughout the training and the discount parameter was fixed. Action choice was made with a probability that is governed by a Boltzman distribution with decaying temperature.

The heuristic agent (HEU) gives a response that is proportional to the percentage of cars in the North-South lane (and its maximum waiting times) relative to the total number of cars (and waiting times).

3.2.2 Advice-exchange

The objective of advice-exchange is to have the agents that are currently reporting better scores help the ones that are doing relatively worst. Agents request advice when the quality of their actions decays to a value lower than a given percentage of the best quality achieved in this problem by any of their peers. The heuristic agent does not participate in the experiments concerning advice-exchange.

Advice-exchange is prohibited in the first 2 to 10 epochs of training. All agents broadcast their best result (i.e., best average quality measured during one epoch) at the beginning of each epoch.

Table 1. Steps of the advice-exchange sequence for an advisee agent (*i*) and an advisor agent (*k*).

1. Agent *i*: receives the best average quality (bq_j) from all other agents ($j \neq i$). Current quality for Agent *i* is cq_i.
2. Loop: Agent *i:* gets state *s* for evaluation.
3. Agent *i:* calculates $k = \arg\max_j(bq_j)$, for all agents ($j \neq i$).
4. Agent *i:* if $cq_i < d \max(bq_k)$, *where* $0<d<1$*:*
   a. Agent *i:* sends agent *k* the current state *s* and requests advice.
   b. Agent *k*: switches to best parameters and runs state *s* to produce its best guess at the adequate response (*g*).
   c. Agent *k*: returns *g* to Agent *i*.
   d. Agent *i*: processes advice (*g*).
5. Agent *i*: runs state *s* and produces response *g'*.

At the beginning of each green-yellow-red cycle, each agent evaluates its current average quality since the beginning of the present epoch. This quality is compared with the best average quality advertised by all other agents at the beginning of the present epoch. If an agents' quality drops below a certain percentage of the best average quality advertised by its peers it will request advice from the agent that has advertised the best average quality. The request for advice is sent having as parameter the current local-state of the environment as seen by the advisee. The advisor switches his working parameters (neural network weights in most cases) to the set of parameters that was used in the epoch where the best average quality was achieved and runs the state communicated by the advisee producing its best guess at what would be the appropriate response to this state. This response (the advised percentage of green time for the North and South lanes in this case) is communicated back to the advisee. In the case where advisees are RW, SA and EA agents, the communicated result is back-propagated as desired response, using the standard backpropagation rule [23] to update the weights of the network immediately after this pass.

When the Q-Learning agent is the advisor, switching to best parameters corresponds simply to select the action with best quality, disregarding the exploration strategy used for learning. In the case where the Q-Learning agent is the advisee, the action that is closest to the given advice (recall that actions are discrete values in this

case) is rewarded using the adaptation equation (1) proposed in Sutton's [37] Dyna algorithm:

$$Q(a \mid s) = Q(a \mid s) + \alpha\,(r + b \sum_{s'} p(s' \mid a, s) \max_{a'} (Q(a' \mid s'))),\ s' \in S_{sa},\ a' \in A_{s'} \qquad (1)$$

where α is the learning rate, r is the estimated reward achieved by taking that action, ß is the discount parameter, $p(s'|a,s)$ is the probability of a transition to state s' given that action a is executed at state s and it is calculated based on previous experience, as the quotient between the number of transitions ($nt_{sas'}$) to state s' when performing action a at the current state, s, and the total number of transitions from current state by action a, i.e.,

$$p(s' \mid a, s) = \frac{nt_{sas'}}{\sum_{i} nt_{sai}}, i \in S_{sa} \qquad (2)$$

where S_{sa} is the set of states reachable from state s by action a, and $A_{s'}$ the set of actions available at state s'. The estimated reward of the advised action will be the average reward advertised by the advisor agent.

After updating its internal parameters with the advised information, the advisee agent gives the appropriate response to the system following the normal procedure for each particular algorithm.

The use of a different type of adaptation when learning from advice, in the Q-Learning agent, is required because the state of the system after the proposed action is unknown. The advisor has not actually performed the action and, possibly, neither will the advisee, so the utility of the next state needs to be estimated by a weighted average of the utilities of all the possible following states when executing the given action.

4 Discussion of the Initial Results

Before starting experiments, some results were expected, namely:

a) After the beginning of advice-exchange, fast, step-like, increases in quality of response. This should happen as soon as one of the agents found a better area of the state space and drove other agents that had worst performances to that area.
b) Faster initial learning rates due to parallel exploration and the sharing of information.
c) Final convergence on better quality values than in tests where no advice is exchanged.
d) Problems of convergence to local optimum when using excess of advice, or high learning rates when processing advice.

In the first set of experiments the parameter that controls the decision of requesting advice to a peer based on the difference between the current quality of the advisee's actions and the best advertised quality, referred as d in Table 1, was set to 1. The con-

sequence of this attribution is that every time an agent has an average quality within the epoch that is lower than the best average quality achieved by its peers in previous epochs, it will request advice. Obviously, this situation happens most of the time, and this has two bad consequences: 1) The time (and resources) required to run the experiments increases several times, compared to other experiments where the value of *d* is lower; 2) As soon as advice-exchange is allowed all algorithms seem to be attracted to the same area of the search-space and even the ones that use random disturbance have some difficulties in leaving this area (because they are very often advised by an algorithm that is fixed to a solution in that area). This is particularly bad since it throws away one of the most promising characteristics of this type of system, which is parallel exploration. It was found that for values of *d* equal or smaller than 0.8 this effect disappears. This was the value used in the experiments reported here.

Table 2 shows a summarized comparison of the main aspects of the training for two groups of agents involved: Stand Alone and Advice-Exchanging agents. Each experiment consisted in two runs with the same initial conditions. In each of these two runs one group controlled the four traffic-lights available. Each set of traffic-lights that control one crossing was controlled by an agent with a different learning strategy. In these experiments the crossings were not interconnected. The results are based on a set of 19 experiments, each running for 1600 epochs, with different parameterizations for the introduction of cars in the beginning of each lane in each experiment. A greater number of experiments was performed, but in some of these experiments learning has stopped within the first 200 epochs. In these experiments one of two situations has occurred: either all the agents have climbed to a good quality value within 200 epochs, easily keeping a regular flow of traffic, or none of the learning agents nor the heuristic have managed to prevent heavy traffic jams. These inconclusive experiments were not considered in the results presented in Table 2.

Table 2. Summary of first results for the traffic-light control experiment for the Stand-Alone and Advice-Exchanging groups.

	Better intial score	Best final score
Stand-Alone	5%	11%
Advice	58%	47%
Tie	37%	42%

The first column shows the percentage of trials in which a group of agents had better scores than its counterparts in the initial phase of the trial. A group of agents is considered a winner if after 500 epochs all its elements are within 10% of the best score achieved so far (regardless of the group whose member has achieved the best score) and if the same condition does not hold for the other group. All other situations are considered a tie.

The second column shows the results for better final score using the same method to determine the winning group.

The actual results observed differ in some respects from expectations. In more than half of the experiments, agents that use advice, do climb much faster to a reasonable

quality plateau in the initial stage of the training, as foreseen in (a), but, contrary to expectations, they do not achieve better final scores as often as expected, only in 21% of the cases a member of the Advise-Exchanging group achieved the best final score. In some cases learning was much slower in the Advice-Exchange group after these fast early increases in quality and the high initial quality value, obtained by advice-exchanging agents, was gradually equaled, or even surpassed, by their Stand-Alone counterparts.

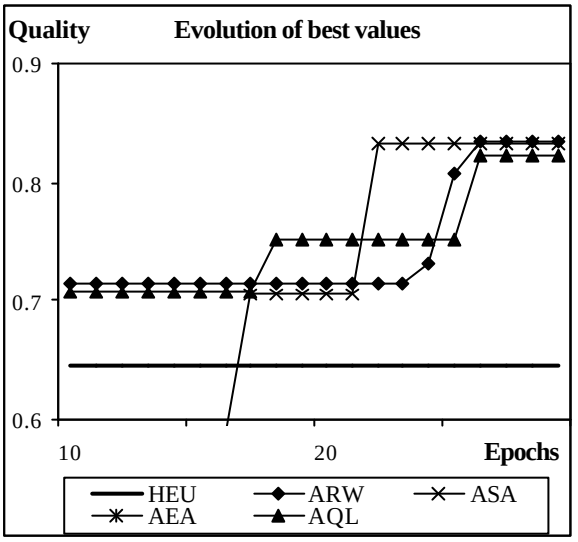

Fig. 1. Detail of the initial phase of a trial. The vertical axis represents the best average quality obtained by each agent up to the moment and the horizontal axis represents the number of epochs elapsed. In this case, advice given by the Q-Learning agent (AQA), first, and Simulated Annealing (ASA), later, led most of the elements in the group on a sudden climb of more than 10% increase in the quality of service. Evolutionary Algorithms also benefited from this jump, but the climb was less steep and from a lower point.

Figure 1 shows a detail of the initial phase of a trial where we can see a typical situation of a fast increase in quality at an early stage caused by advice exchange. The Q-Learning and Simulated Annealing agents jump to higher quality areas and "pull", by advice-exchange, their peers into those areas in a few epochs. In this experiment the advice-exchanging agents did not stop at this quality plateau, being able to obtain better final scores than their counterparts, although learning has slowed from this point on. The Evolutionary Algorithm agent, not seen in the figure, does eventually get to that quality plateau, but it takes longer to show the effects of advice-exchange. This is a common observation and it is due to the distribution of advice by different specimens that slows down the effects of advice-exchange in this type of agents.

One of the most interesting problems observed was that of ill advice. It was observed that some agents, due to a "lucky" initialization and exploration sequence,

never experience very heavy traffic conditions, thus, their best parameters are not suited to deal with this problem. When asked for advice regarding a heavy traffic situation, their advice is not only useless, but also harmful, because it is stamped with the "quality" of an expert. In Q-Learning this was easy to observe because there were situations, far into the trials, for which advice was being given concerning states that had never been visited before. In the next section some measures to prevent this problem will be discussed.

One important characteristic that was observed was that advice-exchanging agents fall into local optima considerably less than their counterparts, being able to cope with bad initial parameters better than the members of the Stand Alone group. This is due to the fact that the supervised learning that occurs based on the advice given by peers helps these algorithms to move into more promising regions of the search-space.

5 Conclusions and Future Work

As mentioned in the previous section, advice-exchange has proved, in these initial trials, to have some advantages over the use of stand-alone agents speeding up the initial learning stages and making the behavior of the group of learners more robust in face of bad initialization. This seems to be a promising way in which agents can benefit from mutual interaction during the learning process. However, this is just the beginning of a search, where a few questions were answered and many were raised. A thorough analysis of the conditions in which this technique is advantageous is still necessary. It is important to discover how this technique performs when agents are not just communicating information about similar learning problems, but attempting to solve the same problem in a common environment. The application of similar methods to other type of learning agents, as well as to other problems, is also an important step in the validation of this approach. It is fundamental to assert to what extent heterogeneity and advice-exchange, each on its own, are responsible for the better performance of a cooperative learning system.

For the time being, experiments in the predator-prey problem and a more realistic traffic environment are under development. This new model for traffic behavior is based on the Nagel-Shreckenberg model, used in [39]. The authors hope that this new formulation provides a richer environment in which advice-exchange can be more thoroughly tested.

One of the main problems observed with advice-exchange is that bad advice, or blind reliance, can hinder the learning process, sometimes beyond recovery. One of the major hopes to deal with this problem is to develop a technique in which advisors and/or advisees can measure the quality of the advice, and agents can develop trust relationships, which would provide a way to filter bad advice. This may be especially interesting if an evaluation of the capability of an agent to advise can be associated with agent-situation pairs. This will allow the advisee to differentiate who is the expert on the particular situation it is facing. Work on the development of "trust" relation-

ships amongst a group of agents has been reported recently in several publications, one of the most interesting for the related subject being [40].

Another interesting issue rises from the fact that humans usually offer unrequested advice for limit situations. Either great new discoveries, or actions that may be harmful for the advisee, seem to be of paramount importance in the use of advice. Rendering unrequested advice at critical points, by showing episodes of limit situations, also seems like a promising approach to improve the skills of a group of learning agents. The same applies to the combination of advice from several sources. These techniques may require an extra level of skills: more elaborate communication and planning capabilities, long-term memory, etc. These capabilities fall more into the realm of symbolic systems. The connection between learning layers, which has been also an interesting and rich topic of research in recent years [41], may play an important role in taking full advantage of some of the concepts outlined above.

Our major aim is to, through a set of experiments, derive some principles and laws under which learning in the Multi-Agent Systems framework proves to be more effective, and inherently different from just having agents learning as individuals.

References

1. M. Tan. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA, 330-337, 1993
2. R. S. Sutton and A. G. Barto. A Temporal-Difference Model of Classical Conditioning. Tech Report GTE Labs. TR87-509.2, 1987
3. S. D. Whitehead. A complexity Analisys of Cooperative Mechanisms in Reinforcement Learning. Proc. of the 9th National Conference on Artificial Inteligence (AAAI-91), 607-613, 1991
4. L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine Learning 8:293-321, 1992
5. C. J. C. H. Watkins, P. D. Dayan. Technical note: Q-learning. Machine Learning 8, 3:279-292, Kluwer Academic publishers, 1992
6. S. D. Whitehead, D. H. Ballard. A study of cooperative mechanisms for faster reinforcement learning. TR 365, Computer Science Department, University of Rochester, 1991
7. M. J. Mataric. Using Communication to Reduce Locality in Distributed Multi-agent learning. Brandeis University Computer Science Technical Report CS-96-190, 1996
8. J. A. Clouse. Learning from an automated training agent. Gerhard Weiß and Sandip Sen, editors, Adaptation and Learning in Multiagent Systems, Springer Verlag, Berlin, 1996
9. R. I. Brafman, M. Tennenholtz. On partially controlled multi-agent systems. Journal of Artificial Intelligence Research, 4:477-507, 1996

10. B. Price, C. Boutilier. Implicit imitation in Multiagent Reinforcement Learning. Proceedings of the Sixteenth International Conference on Machine Learning, pp. 325--334. Bled, SI, 1999
11. H. R. Berenji, D. Vengerov. Advantages of Cooperation Between Reinforcement Learning Agents in Difficult Stochastic Problems. 9th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '00), 2000
12. C. Claus, C. Boutilier. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems
13. S. Kapetanakis, D. Kudenko. Improving on the reinforcement learning of coordination in cooperative multi-agent systems. Symposium on Adaptive Agents and Multi-Agent Systems (AISB/AAMAS-II), Imperial College, London, April 2002
14. C. Baroglio. Teaching by shaping. Proceedings ICML-95. Workshop on Leaming by Induction vs. Learning by Demonstration, Tahoe City, CA, USA, 1995
15. R. Maclin, J. Shavlik. Creating advicetaking reinforcement learners. Machine Leaming 22:251-281, 1997
16. M. J. Mataric. Learning in behaviour-based multi-robot systems: policies, models and other agents. Journal of Cognitive Systems Research 2:81-93, Elsvier, 2001
17. O. C. Jenkins, M. J. Mataric, S. Weber. Primitive-based movement classification for humanoid imitation. Proceedings, first IEEE-RAS international conference on humanoid robotics, Cambridge, MA, MIT, 2000
18. M. Nicoluescu, M. J. Mataric. Learning and interacting in human-robot domains. K. Dautenhahn (Ed.), IEEE Transactions on systems, Man Cybernetics, special issue on Socially Intelligent Agents – The Human In The Loop, 2001
19. M. J. Mataric. Sensory-motor primitives as a basis for imitation: linking perception to action and biology to robotics. C. Nehaniv & K. Dautenhahn (Eds.), Imitation in animals and artifacts, MIT Press, 2001
20. F. J. Provost, D. N. Hennessy. Scaling Up: Distributed Machine Learning with Cooperation. Proceedings of the Thirteenth National Conference on Artificial Intelligence, 1996
21. J. H. Holland. Adaptation in Natural and Artificial Systems. University of Michigan Press, 1975
22. J. R. Koza. Genetic programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge MA, 1992
23. D. E. Rumelhart, G. E. Hinton, R. J. Wlliams. Learning internal representations by error propagation. Parallel Distributed Processing: Exploration in the Microstructure of Cognition, vol. 1: Foundations, 318-362, Cambridge MA: MIT Press, 1986
24. R. Salustowicz. A Genetic Algorithm for the Topological Optimization of Neural Networks. PhD Thesis, Tech. Univ. Berlin, 1995
25. X. Yao. Evolving artificial neural networks. Proceedings of the IEEE, 87(9),1423-1447, 1999
26 A.P. Topchy, O.A. Lebedko, V.V. Miagkikh. Fast learning in multilayered neural networks by means of hybrid evolutionary and gradient algorithms. Proc. of IC on Evolutionary Computation and Its Applications, Moscow, 1996

27. K. W. C. Ku, M. W. Mak. Exploring the effects of Lamarckian and Baldwinian leaming in evolving recurrent neural networks. Proceedings of the IEEE International Conference on Evolutionary Computation, 617-621, 1997.
28. W. Erhard, T. Fink, M. M. Gutzmann, C. Rahn, A. Doering, M. Galicki, The Improvement and Comparison of different Algorithms for Optimizing Neural Networks on the MasPar {MP}-2. Neural Computation {NC}'98, ICSC Academic Press, Ed.M. Heiss, 617-623, 1998
29. P.A. Castillo, J. González, J.J. Merelo, V. Rivas, G. Romero, A. Prieto. SA-Prop: Optimization of Multilayer Perceptron Parameters using Simulated Annealing. IWANN99, 1999
30. T. Hogg, C. P. Williams. Solving the Really Hard problems with Cooperative Search. Proc. 11th National Conference on Artificial Intelligence (AAAI-93), 231—236, 1993
31. C. Goldman, J. Rosenschein. Mutually supervised learning in multi-agent systems. Proceedings of the IJCAI-95 Workshop on Adaptation and Learning in Multi-Agent Systems, Montreal, CA., August 1995
32. T. Thorpe. Vehicle Traffic Light Control Using SARSA. Masters Thesis, Department of Computer Science, Colorado State University, 1997
33. E. Brockfeld, R. Barlovic, A. Schadschneider, M. Schreckenberg. Optimizing Traffic Lights in a Cellular Automaton Model for City Traffic. Physical Review E 64 , 2001
34. L. Nunes, E. Oliveira. On Learning By Exchanging advice. Symposium on Adaptive Agents and Multi-Agent Systems (AISB/AAMAS-II), Imperial College, London, April 2002
35. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi. Optimization by simulated Annealing. Science, Vol. 220:671-680, May 1983
36. M. Glickman, K. Sycara. Evolution of Goal-Directed Behavior Using Limited Information in a Complex Environment. GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, July 1999
37. R. S. Sutton. Reinforcement learning architectures. Proceedings ISKIT'92 International Symposium on Neural Information Processing, Fukuoka, Japan, 1992.
39. K. Nagel, M Shreckenberg. A Cellular Automaton Model for Freeway Traffic. J. Phisique I, 2(12):2221-2229, 1992
40. S. Sen, A. Biswas, S. Debnath. Believing others: Pros and Cons. Proceedings of the Fourth International Conference on Multiagent Systems, 279-286, 2000
41. P. Stone. Layered Learning in Multi-Agent Systems. Tech. Rep. CMU-CS-98-187, December 15, 1998, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1998.