Learning from Multiple Sources

Luís Nunes
ISCTE/FEUP/LIACC-NIAD&R
ISCTE, Av. Forças Armadas, 1649-026 Lisbon, Portugal
Luis.Nunes@iscte.pt

Eugénio Oliveira
FEUP/LIACC-NIAD&R
FEUP, Av. Dr. Roberto Frias, 4200-465, Porto, Portugal
eco@fe.up.pt

Abstract

This work aims at defining and testing a set of techniques that enables agents to use information from several sources during learning. In Multiagent Systems (MAS) it is frequent that several agents need to learn similar concepts in parallel. In this type of environment there are more possibilities for learning than in classical Machine Learning. Exchange of information between teams of agents that are attempting to solve similar problems may increase accuracy and learning speed at the expense of communication. One of the most interesting possibilities to explore is when the agents may have different structures and learning algorithms, thus providing different perspectives on the problems they are facing. In this paper the authors report the results of experiments made in a traffic control simulation with and without exchanging information between learning agents.

1. Introduction

The question we will address is "(How) can agents benefit from exchanging information during learning?" We are particularly interested in situations where learning agents (hereafter referred as agents for the sake of simplicity) may be running different types of algorithms. This work is a continuation of [18, 19] in which the authors presented the first experiments using a set of techniques labelled *Advice-Exchange* (AE). In this work we present several changes to the previous techniques and test them in a Traffic Control simulation. The main differences from previous work lie in the multiple use of feedback information and in the introduction of imitation as another way to use information given by other agents. Agents can also acquire knowledge by learning from known (pre-programmed) heuristics. Even when these heuristics are sub-optimal they can still provide some guidance in the initial stages of learning. This source of information is also explored in the experiments.

The type of environments we are interested in are those where we may have several different teams, each trying to learn the best solution to a given problem. By "learn the best solution" we mean: maximize the reward given by the environment, as defined in equation (1). All problems have the same structure, which means that the actions and the variables that compose a state are the same for all agents. The problems faced by the agents have a stochastic nature and each differs from the others in terms of the initial conditions and the random events generated in each area. The main differences in the dynamics of the problems are, however, a consequence of the interactions between the agents in each team.

An important characteristic of the learning systems under study is that the teams can differ in terms of the type of learning algorithms their members use. In this case we use only two types of learning algorithms: Evolutionary Algorithms (EA) [8, 10] and Q-Learning (QL) [26]. Even though AE is dependent on features that change for each new learning algorithm or problem studied, its main concepts can range a wide variety of learning algorithms and problems. The application of this variant of AE to a problem requires: a reward-function whose result can be scaled to the $[0, 1]$ interval; a function of distance between environment states, whose result is also in (or scalable to) $[0, 1]$.

2. Information Exchange Concepts

To help us envision the concepts that will be defined bellow it is important to have an idea of the type of problem agents are facing. The environment chosen as a testbed is a simplified simulation of a traffic control problem.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'04, July 19-23, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-864-4/04/0007...$5.00

It is important, at this point, to state that we do not propose, that these techniques are an adequate solution to a real-life traffic control problem; even though, some lessons may be taken that apply to this (as well as to other) problem(s) in Multiagent Learning. This test-bed was chosen because it has all of the characteristics of the type of environments these techniques are aimed at. Although the tests are based on real data (graciously made available by the Lisbon City Hall Traffic Department) the constraints put on the simulation, that make it possible to run approximately 1000 times faster than real-time, do not allow us, at this moment, to extrapolate the conclusions taken to a real-life scenario.

In the Traffic Control Simulation (TCS) each team is in charge of two connected crossings (labelled East and West) in a certain *area*. Each crossing is controlled by a different agent. Members of a team may communicate with their *partner* (in the same area) or with members of other teams that are solving similar problems in different areas. In this case we have simulated three different areas running simultaneously in each trial. Since communication protocols and semantics are not an issue in this work it was assumed that these issues will be dealt with outside the scope of this application.

In abstract terms we can say that the problem an agent i has to deal with, at time t, is that of given a *partial* view ($s_{i,t}$) of the global *state* in its area (S_t), choosing an action $a_{i,t}$ that maximizes the average reward ($R_{i,n}$) over a given number of time-steps (hereafter called *epoch*), where n stands for the epoch index. The reward at each time-step $r_{i,t}$, (or, to be precise, $r_{i,t}(s_{i,t+1}, S_{t+1})$) will be the weighted sum of two terms,

$$r_{i,t} = \beta r_l(s_{i,t+1}) + (1-\beta) r_g(S_{t+1}), \qquad (1)$$

the first, $r_l(s_{i,t+1})$, is a measure of the local quality of the state in the vicinity of agent i at time $t+1$, and the second, $r_g(S_{t+1})$, the global quality of the state in the area controlled by agent i's team, after all agents of the team performed their actions. The value of β used was 0.75. This represents the compromise between improving the individual performance at a crossing, versus the global quality in the area controlled by the team. In these experiments the vicinity of an agent is the crossing it controls and the global quality is measured over each area, i.e. all lanes that pass through both crossings controlled by agent i's team.

It is important that these measures of quality are chosen so that the local measure gives, as much as possible, a representation of the agent's contribution to the global performance of the team; but, in some cases, such as the present problem, it is almost impossible to determine exactly how much of the global quality is the responsibility of each given agent. Thus the introduction of a mixed quality measure is necessary to give an agent information about its individual performance as well as its team's.

Each new state of the environment, typically a vector of real numbers, depends on several factors: the previous state; the actions of all agents; random decisions made by the environment (e.g. car acceleration). This type of problems, that combine *partial observability* with a certain degree of randomness, make the problem very difficult to deal with using classical approaches. It is important to notice that, even though the agent receives a reward with a global component, the state it observes ($s_{i,t+1}$) only provides information regarding its own crossing.

2.1. Agent's structure

In this work two different types of agents were used, EA-agents and QL-agents, labelled according to the main learning algorithm used in each case.

Each EA-agent contains a population of Artificial Neural-Networks (ANN) [22] and performs mutation (by adding Gaussian noise with zero average to the weights of the connections and bias). Crossover is done selecting sub-trees randomly from each of the two parents. This process relies mostly on mutation rather than crossover and is in many ways similar to the one described in [7]. Each ANN is evaluated for three epochs. In the first of these information-exchange is inhibited. After this evaluation the specimen's fitness is calculated as the average performance in these three epochs. When all specimens are evaluated, a certain number of them is kept untouched for the next generation (elitist strategy), and the remaining population is generated by mutation and crossover based on a certain number of the best specimens found in the previous generation (approximately $1/3$ of the whole population). The mutation rate is slowly decayed during training.

QL-agents execute a standard Connectionist Q-Learning (ConnQL) algorithm [11]. The purpose is to estimate the infinite discounted quality of each state-action pair $Q(s,a)$. Each agent contains one ANN for each action and upon arrival of reward r_t uses standard Backpropagation [22] to update the quality estimate of a state-action pair, using $r_t + \beta Q_{max}(s_{t+1})$ as the desired output for input s_t over ANN_m, where m denotes the index of the chosen action. The estimated quality of the next state ($Q_{max}(s_t)$), is given by:

$$Q_{max}(s_t) = \max_a(Q(s_t, a)), \qquad (2)$$

for all possible actions a when the system is in state s_t. The parameter β is a discount factor (in this case 0.7). Boltzmann exploration with decaying temperature is used to calculate the probability of each action being chosen. Further details on the basic learning procedures of each agent can be found in the above cited sources and in [20].

One of the most important features of the agents described in this work is that they can learn from advice given

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'04, July 19-23, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-864-4/04/0007...$5.00

by its peers. An *advice* ($adv_{ik,t}$) is composed of a state ($s_{i,t}$), experienced by agent i (hereafter called *advisee*), an action $a_{k,t}$ proposed by agent k (hereafter called *advisor*), the reward, $\hat{r}_{k,t}$, that the advisor would expect to achieve by taking the proposed action and the confidence, $\hat{c}_{k,t}$, the advisor estimates for this information. Advice is generated upon request. When an agent i decides to request advice it will send the observed state to the selected advisor. The advisor will then use a pre-saved set of parameters to generate the proposed action. To estimate the reward that will be obtained, as well as the confidence in the action proposed for this state, the advisor will use information previously saved in an *example store* as will be defined bellow.

Each agent will store three different sets of solution parameters in different *parameter-stores* and the histories of the epochs when those parameters were saved in three *example-stores*. The parameter-store's structure is algorithm dependent. The history of an epoch will be the set of examples observed in that epoch. Each history element (or example) will consist on the state, action, and reward for a given time t ($s_{i,t}, a_{i,t}, r_{i,t}$, respectively), as well as the state observed at time $t+1$, $s_{i,t+1}$. The first pair of stores will contain the parameters of the current hypothesis and the examples observed in the current and last epochs; The second will contain the parameters and examples saved when the best team performance was achieved. The last will contain the parameters and examples used to give advice. This store will be referred to as *advice example-store* and the corresponding parameters as *advice parameters*.

The *advice parameters*, and the corresponding history, are saved when a measure of the *quality of advice* (labelled $q_{k,n}$, for a given advisor k at epoch n) is surpassed. $q_{k,n}$ is updated using the feedback of advisees as well as the performance of the advisor. For a given advisor agent k and advisee agent i, at epoch n, the update of $q_{k,n}$ is the following:

$$q_{k,n+1} = \begin{cases} R_{k,n}, : q_{k,n} < R_{k,n} \\ \alpha q_{k,n} + (1-\alpha) R_{i,n}, : q_{k,n} \geq R_{k,n} \end{cases} \quad (3)$$

with $\alpha \in [0,1]$ (in this case 0.7).

In the first case, (when $q_{k,n} < R_{k,n}$) the advisor achieved an average reward higher than the previous quality of advice, so it will replace the value of $q_{k,n}$ with the one obtained in the current epoch, save a new set of advice parameters and the history of the preceding epoch. In the second case $q_{k,n}$ is updated to reflect the actual value of the advice generated by those parameters for a given advisee i. Several advisees can trigger this update in the same epoch, provided that $q_{k,n} \geq R_{k,n}$ and agent i received advice only from this advisor in epoch n.

From the history of an epoch an agent can extract the estimated reward for a given state, or for a state-action pair. This is done by comparing the state with all the stored cases and fetching the one that is most similar according to a given distance measure, $d(s_1, s_2)$. The distance between states is defined by the environment because it is a problem-dependent feature, the only requirement is that $d(s_1, s_2) \in [0,1]$ for all states s_1 and s_2. In this case we used normalized Euclidean distance. When extracting information for a given state-action pair, the process is similar, except that the agent will only consider examples where the required action was used.

After extracting the best matching example from the advice example-store the agent can use the information on the reward obtained in this case as an estimate of the that which can be achieved using its advice ($\hat{r}_{k,t}$). The confidence ($\hat{c}_{k,t}$) is estimated as $1 - d(s_{i,t}, s_k)$, where $s_{i,t}$ is the state sent by the advisee and s_k is the closest match found in the *advice example-store* of advisor k.

The quality of previous advice given by a certain peer can be seen as a measure of trust. Trust, in this case, is related to how accurate was the estimated reward given for previous advice. When information is requested concerning a given state the advisor will also give the reward that it would expect to achieve using the proposed action. The advisee will initialize trust with 1.0 and update at the end of each epoch (index by n) it in the following way:

$$trust_{n,ik} = \alpha\, trust_{n-1,ik} + (1-\alpha) \sum_t (\hat{r}_{k,t} / r_{i,t}), \quad (4)$$

where $\hat{r}_{k,t}$ is the estimated reward of an advice given at time t by advisor k and $r_{i,t}$ the reward actually received by advisee i when using the advice.

Each agent in a team may know its own role or acquire it in run-time. In this case the roles were statically attributed according to the agent's geographical position in a team (East or West). The role could also have been acquired in run-time by unsupervised learning as was done in previous experiments [20]. The definition of roles will also contribute to a good choice of advisors.

Another important feature, that allows an agent to keep track of its evolution and helps in the process of deciding whether or not to get advice, is the definition of *learning stages*. Agents go through four stages during learning. First, an agent will engage in individual search without communicating with other agents (this was labeled the *Exploration* stage), when the learning stabilizes it will evaluate its performance by comparing it with those of other agents at other areas. It will then jump to one of three stages: *Novice*, *Intermediate* or *Expert*, depending on how good its performance is compared to others'. These stages also help an agent to decide what kind of information is more suitable. If an agent is rated as Novice it will ask for a continuous stream of advice from more experienced agents until one of three conditions is true: a) it reaches a performance level similar to its

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'04, July 19-23, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-864-4/04/0007...$5.00

advisor's; b) concludes that the information from a particular advisor is not adequate for its problem; c) it is unable to learn more from the current advisor, i.e. learning is stabilized. Case b) arises when the performance of an agent drops considerably after starting to get advice from a certain peer. This can happen, even when the advisor is more experienced, due to differences in the partners' strategies. In this case the agent will switch to another, more promising, advisor. In cases a) and c) it will be promoted to Intermediate or Expert stages. When in Intermediate stage an agent knows that it cannot learn much more from a single advisor, but there may be parts of the policies of different agents that can be merged into one better than the sum of the parts. In this stage an agent will ask for advice only for some states in which the choice of the most appropriate action is still unclear or when another agent estimates to achieve a better reward for the current situation. In the Expert stage agents will not ask for advice, dedicating themselves to individual exploration once more and using low learning rates to maintain the team's stability. Different learning stages can also imply different parameterizations of the learning algorithms. A Novice agent will rely more on advice than an Intermediate agent. This can be achieved by using different learning rates in each stage, both for the integration of advised knowledge as well as for the usual process of learning from the environment's reward. The use of learning stages to control learning-rates was previously reported in [6].

2.2. Steps of Advice-Exchange

1	if should get advice($s_{i,t}$) (eq. 5)
2	k = select advisor() (eq. 6 and 7)
3	a'_t = get advice(k, $s_{i,t}$)
4	$a_t = a'_t$
5	a''_t = select action for($s_{i,t}$)
7	else
8	$a_t = a''_t$ = select action for($s_{i,t}$)
9	execute(a_t)

Table 1. Summary of the action selection process, after state $s_{i,t}$ is observed.

Tables 1 and 2 show a brief summary of the main steps of advice-exchange. We will now explain each of these steps in detail introducing the necessary concepts along the way.

When an agent is faced with the decision of choosing an action for a given state it can use its own policy and learn directly from the environment's reward, or ask for the advice of another agent (table 1, line 1). This decision may depend on several factors, such as: a) the experience each agent has in dealing with that state, or similar ones; b) the reward it expects to obtain; c) the quality of previous advice given by its peers; d) the learning stage of the advisee. In the experiments reported bellow an agent will always get advice when in the Novice stage, never get advice in Expert stage, and when in Intermediate stage it will get advice if it finds another agent that has experienced a higher reward with higher confidence when acting from that state, i.e.

$$\begin{aligned} &\exists k \in Agents : k \neq i \wedge \\ &\hat{r}_{i,t}(s_{i,t}) < trust_{n,ik} \cdot \hat{r}_{k,t}(s_{i,t}) \wedge \\ &\hat{c}_{i,t}(s_{i,t}) < \hat{c}_{k,t}(s_{i,t}), \end{aligned} \tag{5}$$

where $\hat{r}_{i,t}(s_{i,t})$ is the estimated reward agent i expects for state $s_{i,t}$, based on the retrieval of the closest state from last-epoch's example-store, and $\hat{c}_{i,t}(s_{i,t})$ is the confidence it has in this estimate. The values for $\hat{r}_{k,t}(s_{i,t})$ and $\hat{c}_{k,t}(s_{i,t})$ are requested to each of the possible advisors, which estimate them based on their advice example-store. Trust is used as weight that maps the average reward announced by each candidate advisor to the actual reward achieved when using its advice.

After an agent has decided to get advice it will need to select an advisor (table 1, line 2). When the advisee is in Novice stage it will only choose an advisor once at the beginning of each epoch. At this stage it is important to keep the same advisor for long periods to prevent conflicting advice. Novice agents are never considered as advisors. A Novice agent i will choose an advisor $k \neq i$ in epoch n, such that:

$$k = argmax_j(trust_{n,ij} \cdot q_{n,j} \cdot rd_j \cdot at_j), \tag{6}$$

for all agents j, where $q_{n,j}$ is the estimated quality of advice, defined in equation (3), given by agent j, rd_j is the role discount and at_j is the current advisor tolerance. The role discount is 1.0 when both agents have the same role and $rd_j \in [0, 1[$, otherwise (in this case we have used a value of 0.9). The advisor tolerance at_j is equal to 1.0 if k is the current advisor and $at_j \in [0, 1[$ otherwise. Again, 0.9 was used in this case. This parameter avoids frequent exchange of advisors that can hurt the learning process.

An agent that is in the Intermediate stage, will have a slightly different way of choosing an advisor. These agents have learned all they can from a single advisor and are, currently, trying to merge policies of different agents to find a better solution to the the problem they are facing. To do this they do not stick to a single advisor; Instead, they try, for each new observed state, to find the best advisor. They will chose a non-Novice advisor $k \neq i$, such that:

$$k = argmax_j(trust_{n,ij} \cdot \hat{r}_j(s_{i,t}) \cdot rd_j), \tag{7}$$

where $\hat{r}_j(s_{i,t})$ is the estimated reward agent j expects for a particular state $s_{i,t}$ (currently observed by advisee i). This

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'04, July 19-23, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-864-4/04/0007...$5.00

estimate is based on the retrieval of the closest match to $s_{i,t}$ from the advice example-store of each candidate advisor j and is communicated to advisor i upon request.

After deciding to which advisor it should ask for help (table 1, line 3) advisee agent i will send it the current state as a parameter of its advice request. The advisor agent k will reload the advice parameters and evaluate state $s_{i,t}$ to produce an advised action (a'_t in table 1, line 3).

When the advised action a'_t is received the advisee faces another decision, it can either update its policy by incorporating the knowledge sent by the advisor, and then evaluate state $s_{i,t}$ with the new policy, or imitate the received action and then observe the consequences. In previous experiments [19] we have focused on immediate integration of advice prior to the action selection, currently we are studying the effects of imitation (with integration of knowledge after the reward is received), so in this case agents will always chose imitation in this stage and use the proposed action directly.

At this point the agent will issue an action to the environment that will either be the advised action, or, when no advice was requested, an action of its own choice. It is important to notice that the advised action and the one agent i would choose on its own may be the same (i.e. in table 1, $a'_t = a''_t$).

1	if advised
2	reward advice($r_{i,t}$)
3	learn from imitation($s_{i,t}$, a_t, $s_{i,t+1}$, $r_{i,t}$)
4	if ($a_t = a''_t$)
5	learn($s_{i,t}$, a_t, $s_{i,t+1}$, $r_{i,t}$)

Table 2. Summary of the credit attribution step, after receiving reward $r_{i,t}$.

After considering all actions from the different agents and calculating its consequences the environment will send each agent its reward (equations 1, 8 and 9). When receiving reward $r_{i,t}$, agent i will verify if it was advised (table 2, line 1). When this is the case it will do the following steps: 1) calculate the product $(\hat{r}_{k,t}/r_{i,t})$ as contribution to the update of $trust_{n,ik}$ (equation 4); 2) learn from the result of imitating the advised action (table 2, lines 3 and 5). In this last step (2), EA and QL-agents must behave differently. QL-agents can learn from the case history regardless if the executed action is the same they would choose or not, but for EA-agents this is not true. EA-agents cannot reward a specimen for an action it did not choose, otherwise it would falsify the specimen's fitness, but the information acquired by using the advice can still be used to train the ANN that corresponds to the active specimen. The EA-agent will use, standard, on-line, backpropagation with $s_{i,t}$ as input and the advised action (a'_t) as desired output. EA-agents will also store the advised action along with the observed state for future replay. At the end of each epoch Novice EA-agents will replay the stored advice for a predefined number of epochs (10 in this case), again using backpropagation in the same way as explained above. Stored advice is cleared when the agent switches to a different advisor.

3. Experiments

In previous experiments different versions of Advice-Exchange were tested in the Pursuit Problem [18, 19]. In the experiments reported bellow we used a Traffic Control simulation (TCS). We will, in this section, describe the experiment skipping some details due to space constraints. A fully detailed description of the experiment is available in [20].

As was mentioned above the experiments are based on real data. The raw data contains the number of cars passing in some of the most busy avenues of Lisbon in each 5 minutes period, during 15 consecutive days. Based on this data the environment generates the number of cars that must enter each lane every 60 seconds to produce the same throughput. To avoid having epochs that are relatively easy to solve, and produce erroneous estimates of an agent's performance, we have discarded all data from weekends and nights (from 8 p.m. to 8 a.m.). Each environment contains three areas, in each area a team of agents controls two connected crossings, each with two incoming lanes in each direction (North, South, East and West) in a total of eight lanes per crossing. Each team uses a different decision mechanism, team 1 is composed of EA-agents, team 2 of QL-agents. A third team, of Heuristic agents (H-agents), performs a fixed, hand-coded, policy. H-agents respond to occupation thresholds, changing to green the traffic-lights of lanes that have higher occupation rates. To avoid quick oscillation the lanes that are currently green have a tolerance bonus that allows them to keep the traffic light open even when their occupation is slightly less than in the other lanes. H-agents do not request advice, but they respond to all advice requests.

Cars are inserted in platoons every 60 seconds, simulating fixed timing traffic-lights outside of the environment's scope. Cars acceleration is calculated as defined in [16], using a random term, to simulate real drivers' reactions. Due to computational constraints car movements have the following limitations: 1) cars do not change lane or turn, 2) forward movement only is limited by: the position of the car directly ahead; current speed; the desired maximum speed of each car; existence of a closed traffic-light ahead; 3) cars can break to zero-speed instantly; 4) stopped cars, in crossings, do not prevent others from passing in other directions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'04, July 19-23, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-864-4/04/0007...$5.00

The simulation is updated every second calculating the movement of all cars within scope. Traffic-lights are prompted for a new decision every 20 seconds. Each decision consists on choosing the color of the light (red or green) for the North-South lanes for the next 20 seconds. The color of the light for the East-West lanes changes automatically to the opposite color. When a light is changed from green to red a yellow light period of 5 seconds is automatically introduced. An epoch, consists of 180 decisions, which is equivalent to one hour of simulated time. This simulation runs 10000 epochs in training mode (in 6 to 10 hours, approximately 1000 times real time). After this, the best parameters saved are reloaded and it runs 500 epochs in test mode (without, communication and learning).

The state that agents observe is composed of 10 real parameters all in $[0, 1]$. Parameters 1 to 4 contain the normalized occupation rate in each of the four incoming directions, i.e. number of cars present in the most busy lane in a given direction (North, South, East or West) over the number of cars required to fill the smallest lane in the area. Parameters 5 to 8 are set to 1 if there is incoming traffic from a given direction (and 0 otherwise). It is considered that there is incoming traffic if a car has entered the lane in the last 10 seconds or if the lane is full. Parameter 9 represents the current color of the agent's traffic-light (0 for red, 1 for green). The last parameter, is the time since the last change in the traffic-light, normalized at a value of 180 seconds.

Before each new decision agents are given a reward, structured as explained in equation (1), where

$$r_l(s_{i,t}) = p_i \cdot C_l / ast_{i,t} \tag{8}$$

$$r_g(S_t) = p_m \cdot C_g / atc_{i,t} \tag{9}$$

where C_l is the patience threshold of drivers for one crossing (10 seconds in this case), $ast_{i,t}$ is the *average stopped time* of cars currently situated in the incoming lanes of the crossing controlled by agent i. If $ast_{i,t}$ is smaller than C_l it takes the value of C_l so that the first component is always in the interval $[0, 1]$. p_i is a penalty value that is 1.0 if all incoming lanes have an occupation-rate smaller than 1.0, and is multiplied by a fixed value (0.75 in this case) for every full incoming lane. The calculation of the global component is similar, although it uses a different patience threshold ($C_g = 20$ seconds) and $atc_{m,t}$ is the average stopped time for all cars in area m (controlled by agent i and its partner). This value includes the time cars may have waited before entering the scenario. The global penalty value (p_m) is computed as p_i, but considering all lanes in area m, even those not controlled by agent i.

4. Results

The results presented in tables 3 and 4 refer to the TCS2 scenario (TCS with 2 connected crossings). In table 3, the column labelled "Avg", contains the average results in test for agents of a given type in *social* trials (with communication) and *individual* trials (without communication). Under the label "Std. Dev." are the standard deviations of each of the values for average performance in 11 trials. The column labelled "Best" shows the average test result of the best agent. The column labeled % shows the percentage relative to the best average result. Table 4 presents the average team performances (also in test) in experiment TCS2 and is organized in the same fashion as table 3. The label "HE" stands for H-agents.

Scenario	Alg	Avg	%	Std. Dev.	Best
Individual	HE	0.2708	93%	+/-0.0013	0.27
Individual	QL	0.2852	98%	+/-0.0093	0.30
Individual	EA	0.2638	91%	+/-0.0347	0.37
Social	HE	0.2710	93%	+/-0.0011	0.27
Social	QL	0.2895	100%	+/-0.0111	0.31
Social	EA	0.2859	98%	+/-0.0352	0.40

Table 3. Test results in experiment TCS2 for each scenario-algorithm pair.

In the TCS2 tests (table 3), the average results of the best agents (QL) are similar in individual and social trials, but EA-agents show a small (7%) improvement in performance. EA-agents seem able to reach scores higher than QL-agents as can be seen in the Best results column. The best EA-agent's performance is increased even when none of their peers is able to reach scores at the same level. This seems to indicate that, in some cases, communication can lead a agents to behave better is social domains than the best they are able to do in any individual trial. Nevertheless, in average we can only say that the agents with lower individual performance (EA-agents) can learn to behave as the best (with only 2% difference to QL-agents, which, given the standard deviations, is not meaningful).

If we measure only the global component of the reward (table 4) the improvements are more obvious. EA-agents improve this component by 22%, in average. QL-agents do not fully take advantage of this rise, but they also show a small improvement (4%). It is interesting to notice that the agents with worse team performance in individual trials rise to the best performance in social ones, so they have not only taken advantage of the knowledge of others, but they seem to have improved it.

These tests seem to indicate that advice exchange does improve the average performance of the agents with lower

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'04, July 19-23, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-864-4/04/0007...$5.00

Scenario	Team	Avg	%	Std. Dev.	Best
Individual	HE	0.1041	94%	+/-0.0011	0.11
Individual	QL	0.1024	92%	+/-0.0089	0.12
Individual	EA	0.0865	78%	+/-0.0247	0.16
Social	HE	0.1044	94%	+/-0.0007	0.11
Social	QL	0.1066	96%	+/-0.0100	0.13
Social	EA	0.1106	100%	+/-0.0274	0.18

Table 4. Global reward results in experiment TCS2.

scores (both individual and team scores of EA-agents), although not all agents reach the same performance levels and some advisees do not reach the level of performance of the advisors. The rise in QL-agents' performance is not significant.

5. Related Work

The work on information exchange between QL-agents, started in the early nineties. In [27], Whitehead tested a cooperative learning architecture: *Learning By Watching* (LBW). In LBW, the agent learns by watching its peers' behavior (which is equivalent to sharing series of state, action, quality triplets). In LBW an agent will use other agents' episodes as if they were its own.

The results presented in [5, 3] are reviewed and expanded in Clouse's Ph.D. thesis [4]. This important contribution reports the results of a strategy labeled *Ask for Help* (AH), in which QL-agents learn by asking other agents of the same type for suggestions and perform the suggested actions.

The work of Lin [11] uses an expert trainer to teach lessons to a QL-agent that is starting its own training. In [25], Tan addressed the problem of exchanging information between QL-agents during the learning process. This work reports the results of sharing several types of information amongst agents that are working in the pursuit (predator-prey) problem. Experiments were conducted in which QL-agents shared policies (internal solution parameters), episodes (series of state, action, quality triplets), and sensation (observed states).

Also related to the work presented bellow, is the research on trust relationships between agents, mainly developed by Sen's research group [23, 2, 1]. This series of works, and other related papers by the same authors, focus on several aspects of learning to trust other agents

Several researchers have been studying applications of Multiagent Reinforcement Learning to stochastic-games. The first references to this concept can be found in [12]. The most recent developments in this research area appear in [9], where a different update rule for joint action Q-Learners was proposed, labelled *Nash Q-Learning*. In this case the update of the quality values is made under the assumption that all other agents will adopt an equilibrium strategy in all future actions.

Incorporation of human advice in reinforcement learners was also studied from several different perspectives [13, 14, 15, 17].

In [24] the *"Agent Teaching Agent"* framework relies on an expert to teach a student agent a certain concept by selecting the training examples to use on each epoch based on the errors the student has made in the previous epochs.

Training of novice Q-learning agents by *implicit imitation* of other pre-trained expert-agents has been studied and successfully applied by Price [21]. Contrary to our study, *implicit imitation* assumes self-interested agents that make no effort to aid others. Agents take advantage solely of watching the results of others' behaviors.

6. Conclusions and Future Work

The objective of this work is to make a contribution in answering the question "(How) can heterogeneous groups of agents benefit from communication to improve their learning skills?". In this step of the way we have proposed and tested an integrated set of techniques that has improved the average skills of learning agents in a traffic-control simulation. One of our major expectations for this work was that an essentially different type of learning (and its results) could emerge from the application of simple rules for exchange of information during learning. It was expected that the use of different learning algorithms, each with a different way of exploring the solutions' space, would increase the likelihood of agents helping others to climb out of local optima. So far the results are positive, although beneath our initial expectations. It is, nevertheless, interesting to observe that the most efficient agents regularly appear in social environments. Despite this fact the average results do not allow us to conclude that a social environment will, with high probability, generate performances that are significantly better than the best that can be achieved by individual learning. The common fact in these as well as other previous experiments is the rise in the avergae performance of agents that have lower scores in individual trials.

It is reasonable to argue that the improvements were made at the expense of computational power and communication bandwidth which is not accounted for. This is true, but despite the need for extra computational power the system still runs hundreds of times faster than necessary for a real application of this type, and communication, although bursty, is, in average, low. Nevertheless, it is important in the future to measure how much more time and bandwidth it takes to exchange information during learning.

The future work agenda also contains the following topics: influencing partners; team supervisors that relate sum-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'04, July 19-23, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-864-4/04/0007...$5.00

marized views of the state to combined actions, and influence teammates to coordinate their actions; a study of how the different parameters (like role discount, last advisor tolerance, individual/global reward balance, etc.) can affect the results; the extension of the concept to other learning algorithms and problems.

Acknowledgements

The authors wish to thank Lisbon's City Hall Traffic Department for graciously making available the necessary data to design the experiments and three anonymous reviewers for their helpful comments.

References

[1] B. Banerjee, R. Mukherjee, and S. Sen. Learning mutual trust. In *Working Notes of AGENTS-00 Workshop on Deception,Fraud and Trust in Agent Societies*, pages 9–14, 2000.

[2] A. Biswas, S. Sen, and S. Debnath. Limiting deception in groups of social agents. *Applied Artificial Intelligence Journal, Special Issue on Deception, Fraud and Trust in Agent's Societies*, 14(8):785–797, 2000.

[3] J. Clouse and P. Utgoff. A teaching method for reinforcement learning. In *Proc. of the Nineth International Conf. on Machine Learning*, pages 92–101, 1992. Aberdeen, Scotland.

[4] J. A. Clouse. *On integrating apprentice learning and reinforcement learning*. PhD thesis, University of Massachusetts, Department of Computer Science, 1997.

[5] J. A. Clouse and P. E. Utgoff. Two kinds of training information for evaluation function learning. In *Proc. of AAAI'91*, 1991. Anaheim.

[6] M. Dorigo and M. Colombetti. Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, 71(2):321–370, 1994.

[7] M. Glickman and K. Sycara. Evolution of goal-directed behavior using limited information in a complex environment. In *Proc. of the Genetic and Evolutionary Computation Conf., (GECCO-99)*, 1999.

[8] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[9] J. Hu and M. P. Wellman. Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 2003.

[10] J. R. Koza. *Genetic programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge MA, 1992.

[11] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992.

[12] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. of the 11th International Conf. on Machine Learning*, pages 157–163, 1994.

[13] R. Maclin and J. Shavlik. Incorporating advice into agents that learn from reinforcement. In *Proc. of the Twelfth National Conf. on Artificial Intelligence (AAAI-94)*. AAAI Press, 1994.

[14] R. Maclin and J. Shavlik. Creating advicetaking reinforcement learners. *Machine Learning*, 22:251–281, 1996.

[15] M. J. Matarić. Using communication to reduce locality in distributed multi-agent learning. Computer Science Technical Report CS-96-190, Brandeis University, 1996.

[16] K. Nagel and M. Shreckenberg. A cellular automaton model for freeway traffic. *J. Phisique I*, 2(12):2221–2229, 1992.

[17] M. Nicolescu and M. J. Matarić. Learning and interacting in human-robot domains. *IEEE Transactions on systems, Man and Cybernetics, Special issue on Socially Intelligent Agents - The Human In The Loop*, 2001.

[18] L. Nunes and E. Oliveira. On learning by exchanging advice. In *Proc. of the First Symposium on Adaptive Agents and Multi-Agent Systems (AISB'02)*, April 2002. Imperial College, London.

[19] L. Nunes and E. Oliveira. Cooperative learning using advice exchange. *Adaptive Agents and Multiagent Systems, Lecture Notes in Computer Science, Hot Topics*, 2636, April 2003.

[20] L. Nunes and E. Oliveira. Exchange of information during learning. Technical Report 3-02/04, FEUP/LIACC, 2004. unpublished, http://home.iscte.pt/˜lmmn/www/publications.htm.

[21] B. Price and C. Boutilier. Implicit imitation in multiagent reinforcement learning. In *Proc. of the Sixteenth International Conf. on Machine Learning*, pages 325–334, 1999.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, 1:318–362, 1986.

[23] S. Sen. Reciprocity: a foundational principle for promoting cooperative behavior among self-interested agents. In *Proc. of the Second International Conf. on Multiagent Systems*, pages 322–329. AAAI Press, Menlo Park, CA, 1996.

[24] S. Sen and P. P. Kar. Sharing a concept. In *Working Notes of the AAAI-02 Spring Symposium on Collaborative Learning Agents (AAAI Tech Report SS-02-02), in Stanford, CA*, 2002.

[25] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proc. of the Tenth International Conf. on Machine Learning*, pages 330–337, 1993.

[26] C. J. C. H. Watkins and P. D. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[27] S. D. Whitehead. A complexity analisys of cooperative mechanisms in reinforcement learning. *Proc. of the 9th National Conf. on Artificial Inteligence (AAAI-91)*, pages 607–613, 1991.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'04, July 19-23, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-864-4/04/0007...$5.00