Human Guided Ensemble Learning in StarCraft

Timothy Verstraeten* Roxana Rădulescu* Yannick Jadoul
Tom Jaspers Robrecht Conjaerts Tim Brys
Anna Harutyunyan Peter Vrancx Ann Nowé

tiverstr,rradules@vub.ac.be
Vrije Universiteit Brussel
Pleinlaan 2
1050 Elsene

ABSTRACT

In reinforcement learning, agents are typically only rewarded based on the task requirements. However, in complex environments, such reward schemes are not informative enough to efficiently learn the optimal strategy. Previous literature shows that feedback from multiple humans could be an effective and robust approach to guide the agent towards its goal. However, this feedback is often too complex to specify beforehand and should generally be given *during* the learning process. We introduce real-time human guided ensemble learning, in which feedback from multiple advisers is learned simultaneously with the agent's behaviour. We evaluate our approach in a small scale one-on-one combat scenario in StarCraft: Brood War. Our preliminary results show that a single expert adviser can provide proper guidance, while using groups of multiple advisers does not improve the convergence speed. In future work, we will investigate an alternative to the tile-coding approximator in order to effectively incorporate advice from multiple humans.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

Keywords

human advice, reward shaping, reinforcement learning, ensemble learning

1. INTRODUCTION

Reinforcement learning (RL) [**?**] allows goal-oriented agents that are able to sense and act upon their surroundings to learn an optimal control-policy from scratch just by interacting with the environment. Despite offering powerful learning mechanisms, one major drawback of RL stands out: an impractical large amount of experience is necessary to reach a good solution. Speeding up the learning process has thus become a focus point in RL research. A few developed directions include learning by demonstration to acquire a good initial policy [**?**, **?**], transfer learning from another related RL task [**?**] and providing additional guidance to the RL agent [**?**].

We focus here on the latter approach, namely reward shaping (RS), through which the agent can receive a more informative reward after each action taken, thus enabling it to converge faster towards the optimal policy. This additional guidance during the learning process can be obtained by specifying knowledge manually into the RS function [**?**] or by learning feedback from a human adviser as a suitable RS function [**?**], an approach that is considered in the current work.

While leveraging the human problem solving capacity can provide an immeasurable benefit, one has to consider handling the mistakes that humans can make during the feedback process. One possible approach consists in building an ensemble system that can robustly handle guidance from multiple sources, in order to avoid the unreliability issue arising from only one [**?**].

We introduce human ensemble learning, a mechanism which aggregates a set of learners, each of them guided by a human adviser. Related work already provides frameworks to combine the advice from multiple external sources [**?**, **?**]. In these cases, human feedback is considered either as a separate reward function, independent but closely related to the actual reward function imposed by the environment, or as a priorly known expert advice function. However, in most settings, human advice is secondary to the feedback from the environment with as sole purpose to guide the agent. Additionally, behavioural advice is in general too complicated to manually specify. We provide a framework to *learn* multiple human advice functions and incorporate them as reward shapings to guide the ensemble agent towards the optimal strategy.

As a testing scenario we have chosen StarCraft,[1] a Real-Time Strategy (RTS) game in which complex combat strategies have to be employed in order to win the game. While RTSs have proven to be a challenge for AI [**?**, **?**], humans seem to succeed in devising different strategies and solutions to achieve victory.

Outline. We start by explaining the components of our human guided ensemble learning framework in Sections 2, 3 and 4. Section 5 describes our experimental setup, while in Section 6 we discuss our preliminary results. We conclude our work in Section 7.

*These authors contributed equally to this work.

[1]Created by Blizzard Entertainment: `http://blizzard.com/games/sc/`

2. REINFORCEMENT LEARNING

In reinforcement learning, an agent learns the policy that maximizes the cumulative reward for achieving a certain goal over a sequence of observations at time-steps $t \in \mathbb{N}$. Based on its current policy and environment, the agent executes an action and updates its policy based on the reward given by the environment. This environment is modelled as a Markov decision process (MDP) $M = (S, A, T, \gamma, R)$ [**?**], where S, A are the state and action spaces, $T\colon S \times A \times S \to [0, 1]$ is a probabilistic transition function, γ is a discount factor determining the importance of future rewards and $R\colon S \times A \times S \to \mathbb{R}$ is the immediate reward function.

An agent behaves according to a policy $\pi : S \times A \to [0, 1]$, meaning that in a given state, actions are selected according to a certain probability distribution. Optimizing π is equivalent to maximizing the expected discounted long-term reward from a given starting state s_0 and action a_0:

$$Q^{\pi}(s, a) = \mathbb{E}_{T,\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s = s_0, a = a_0\right] \quad (1)$$

where r_{t+1} is the reward obtained upon taking action a_t at state s_t. This is the idea of value-based methods, in which a value function (VF) maintains and optimizes iteratively the expected long-term reward for each state-action pair based on the observations made during explorations of the environment.

A method of this kind is SARSA, which updates its *Q-values* using the temporal-difference (TD) δ between subsequent state-action pairs (s, a) and (s', a'):

$$Q^{\pi}(s, a) = Q^{\pi}(s, a) + \alpha\delta \quad (2)$$

$$\delta = r_{t+1} + \gamma Q^{\pi}(s', a') - Q^{\pi}(s, a) \quad (3)$$

where α is the learning rate, and s' and a' are taken according to respectively T and π. This approach falls into the category of on-policy methods, which means the agent will learn the value function of its behaviour policy.

In general, the *target* policy can be different from the behaviour, in which case, the learning is off-policy. For example, Q-learning computes the value function of the greedy policy π^*. The TD update rule is replaced by:

$$Q^{\pi^*}(s, a) = Q^{\pi^*}(s, a) + \alpha\delta$$

$$\delta = r_{t+1} + \gamma \max_{a'} Q^{\pi^*}(s', a') - Q^{\pi^*}(s, a)$$

where the TD is computed w.r.t. the greedy action a'.

3. REWARD SHAPING

In order to speed up convergence towards the optimal policy, guidance can be provided to the agent by augmenting the reward function R with an additional more informative reward shaping function F.

It is necessary and sufficient to consider F as a potential-based reward shaping function (PBRS) in order to guarantee policy invariance and make sure that the shaping does not lead to learning sub-optimal policies [**?**]. A PBRS function is constrained by a real-valued potential function Φ as follows:

$$F(s, s') = \gamma\Phi(s') - \Phi(s)$$

We can extend F further by including behavioural knowledge, giving us an advice PBRS function [**?**].

$$F(s, a, s', a') = \gamma\Phi(s', a') - \Phi(s, a) \quad (4)$$

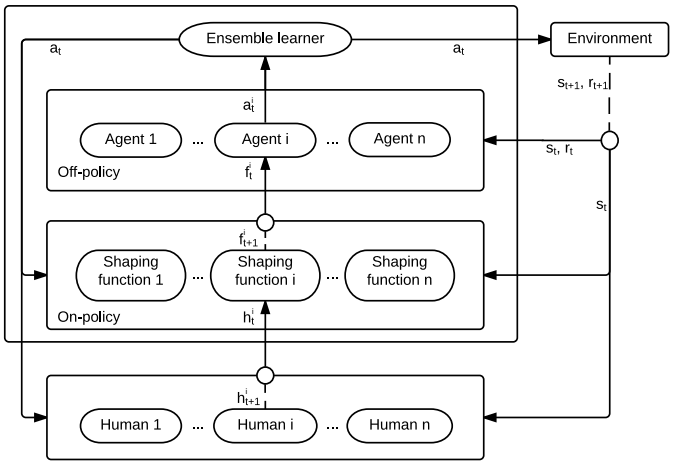


Figure 1: Hierarchy of ensemble system – The ensemble agent executes an action a_t based on the preferences $p^i(s_t, a_t^j)$ for every possible action a_t^j of each of its off-policy sub-agents i. The sub-agents update their Q-values based on a reward signal and the next state given by the environment. This reward signal is shaped for each agent by f_{t+1}^i, a value provided by an advice PBRS function. The advice function captures the intended feedback of human adviser i, given his reward signals h_{t+1}^i.

Inserting such knowledge manually under the constraints of a potential function Φ can prove to be difficult. However, a framework can be constructed to incorporate external guidance, e.g., by a human expert, as a secondary VF [**?**]. This additional VF specifies the values of the advice policy intended by the human. More specifically, it defines the discounted future rewards given by the adviser at each state-action, based on the immediate rewards provided by the human after each action (i.e., fixed positive rewards for encouraged actions and zero otherwise).

The agent learns this advice policy simultaneously with its own behaviour (defined by the primary VF). In order to guarantee convergence towards the intended advice function, the secondary VF is learned on-policy [**?**].

This VF can then be used as a potential function in Equation 4 in order to shape the rewards given by the environment. Combining this advice PBRS function with Equation 3, we have:

$$\delta = r_{t+1} + f_{t+1} + \gamma Q(s', a') - Q(s, a)$$

$$f_{t+1} = \gamma\Phi_{t+1}(s', a') - \Phi_t(s, a)$$

where Φ_t is the secondary VF at time t.

4. HUMAN GUIDED ENSEMBLE LEARNING

In ensemble RL, an agent manages multiple sub-agents j and aggregates their individually learned policies π_j in order to define one ensemble policy π [**?**, **?**]. One way to combine the strategies of the sub-agents is to compute a preference value $p(s, a)$ for each action a in state s of the ensemble agent by aggregating the preference values of the sub-agents. An example of this is rank voting (RV) [**?**], in

which each sub-agent j assigns a rank $r^j(s, a_i) \in [1, \cdots, n]$ to all its n actions a_i in state s, such that:

$$r^j(s, a_1) > r^j(s, a_2) \Leftrightarrow Q^{\pi_j}(s, a_1) > Q^{\pi_j}(s, a_2)$$

Ties in the Q-values result in the same rank for both actions.

The preference values of the ensemble agent are then described as follows:

$$p(s, a) = \sum_{j=1}^{m} r^j(s, a)$$

An exploratory policy (such as ϵ-greedy [**?**]) can be established over these preference values to define the behaviour of the learning ensemble agent.

Using the combination of ensemble learning and reward shaping by an external human adviser, we can now describe a framework that is capable of aggregating feedback, given in parallel by multiple human advisers, in order to offer a more robust guiding strategy for an RL agent. Every human guides a single sub-agent. This means that the ensemble agent manages sub-agents observing the same experience, but with differently shaped Q-values.

With the aim of maximizing the diversity over the preferences of the sub-agents, we maintain an ensemble of Q-learning agents (i.e., off-policy learners), as this does not include learning the similar exploratory behaviour of the agents. However, the secondary VFs (i.e., the potential functions Φ, used to define the advice PBRS function in Equation 4) should be learned on-policy [**?**]. An overview of the ensemble system is given in Figure 1.

5. EXPERIMENTS

We evaluate our human guided ensemble learning framework in the context of a one-on-one combat scenario in StarCraft. We investigate whether incorporating feedback from multiple humans yields faster convergence towards the optimal policy.

5.1 StarCraft Environment

We assess our approach in the context of StarCraft: Brood War, a Real-Time Strategy (RTS) video game. As a reinforcement learning framework, we employ Brood War API (BWAPI), which is an open-source library that allows scripted interaction with the StarCraft environment [**?**]. It has been used in RL research as an experimental setting for both small scale combat [**?**] as well as more complex scenarios [**?**].

We focus on training an agent for a small scale one-on-one combat in a setting inspired by [**?**]. This allows us to study the performance of our ensemble agent, solving a rather simple task in a complex, yet entertaining, environment.[2]

The state space of the StarCraft environment consists of the following features: the position of the agent in the continuous xy-coordinate system (with $x, y \in [0, 1000]$), the absolute vector from the agent to the enemy w.r.t. the coordinate system, the difference in health points (HP) and whether or not the enemy is engaging in combat. An agent can move in all cardinal directions over a fixed distance and is allowed to engage the enemy, move towards the enemy or stay idle. Additionally, the agent can shoot from a certain distance, while the enemy can only attack in close range.

[2]The source code is available at `https://github.com/timo-verstraeten/human-ensembles-starcraft`

The state space is discretized by employing a tile-coding function approximator [**?**], such that each feature has 4 tilings. This implies that the state space is *not jointly* discretized over all features, and thus features are considered independently in the computation of a Q-value.

The goal of the agent is to kill the enemy using a minimal number of steps, while having left as many health points as possible. The agent receives rewards according to the following formula:

$$R(s, s') = \begin{cases} \mathrm{HP}_{agent} - \mathrm{HP}_{enemy}, & \text{if a player dies} \\ -0.3, & \text{otherwise} \end{cases}$$

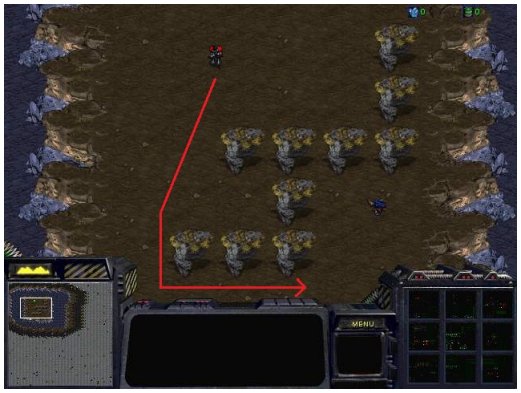
where $\mathrm{HP}_{agent} \in [0, 20]$ and $\mathrm{HP}_{enemy} \in [0, 35]$ are respectively the agent's and enemy's HP. The damage done by each character was set in such a way that close-range combat would kill the agent. Thus, a more complex strategy than 'rushing' towards the enemy should be employed by the agent in order to win.

5.2 Experimental Setup

For each experiment, we assembled unique groups of advisers, each consisting of five people. During each experiment, the agent receives advice simultaneously from all the members of the group. Each individual was isolated and only had the real-time visual information as it is given in the original StarCraft environment. Thus, no information about the underlying reinforcement learning problem was available (such as Q-values). Prior to the experiments, they were also informed about optimal and suboptimal strategies that can be employed in the combat scenario, shown in Figure 2, and the possible actions the agent can take, to ensure that each human has the knowledge to be a proper adviser. They could communicate their feedback in the form of a binary signal (i.e., they provide a positive reward whenever they want to endorse the agent's action) during the first 5 episodes of the agent's learning process. We have chosen to only allow positive feedback signals, as learning sparse and all-positive (or all-negative) feedback captures the intended advice more robustly [**?**]. We made sure that the advisers had enough time to provide their feedback on the current action by slowing down the game speed. After the first 5 episodes, the agent had to learn the optimal behaviour on its own for 195 additional episodes. An episode terminates when one of the characters is defeated. Additionally, after 1000 steps, the agent rushes towards the enemy in order to cut-off the episode.

As explained before, the optimal policies of the sub-agents are learned off-policy, while the secondary VFs are learned on-policy. We respectively used Q(λ) and SARSA(λ), which both employ *eligibility traces* in order to update previously encountered Q-values with an impact factor of λ, using the currently observed reward [**?**].

The ensemble agent uses RV to choose an action based on the greedy actions of the sub-agents. Following an ϵ-greedy policy ($\epsilon = 0.1$), the ensemble agent can execute this action or alternatively select a random action with a probability of ϵ. The discount factor γ is 1.0 in our problem setting. For the learning rate α and eligibility traces decay factor λ for our primary VF, we respectively use 0.40 and 0.95, which are jointly optimized for an ϵ-greedy Q(λ) agent (using the same ϵ and γ as mentioned before). Additionally, we took the number of tilings (of the CMAC function approximator) to be 4. The tile resolutions for the distance and angle features



(a) Kamikaze strategy – The agent rushes towards the enemy in an attempt to minimize the number of steps (local optimum).

(b) Optimal strategy – The agents shoots from behind the trees, where the enemy cannot reach.

Figure 2: Possible strategies in the StarCraft scenario

(defining the vector between the agent and enemy) were set to 30 and 10, while the resolution for the health feature was set to 0.7. These tiling parameters are chosen in such a way that they generalize the state space well, such that there is a noticeable impact of the human advice, while still providing good results. The positive reward provided by the human advisers is set to 10, as this gives us the best results. We alter γ and α used in the human advice potentials to 0.5 and 0.6 to ensure faster convergence in the secondary VFs.

6. RESULTS AND DISCUSSION

We first present results for guidance by a single expert, in comparison with an ensemble of five expert advisers. We analyse the effect of the function approximator under human advice on the results. We then study the results obtained by introducing non-expert advisers, comparing their performance and advice frequencies to the ensemble of experts.

6.1 Single Expert vs Multiple Experts

We show results for two groups which are new to the advising scene, and a group of experts (i.e., the first five authors of this paper) who know the underlying state-action space, are familiar with RL and had a lot of individual practice with offering feedback to StarCraft agents. They can practice the feedback mechanism for 1 test trial, after which they do 9 actual trials. These 9 trials are incorporated in the results.

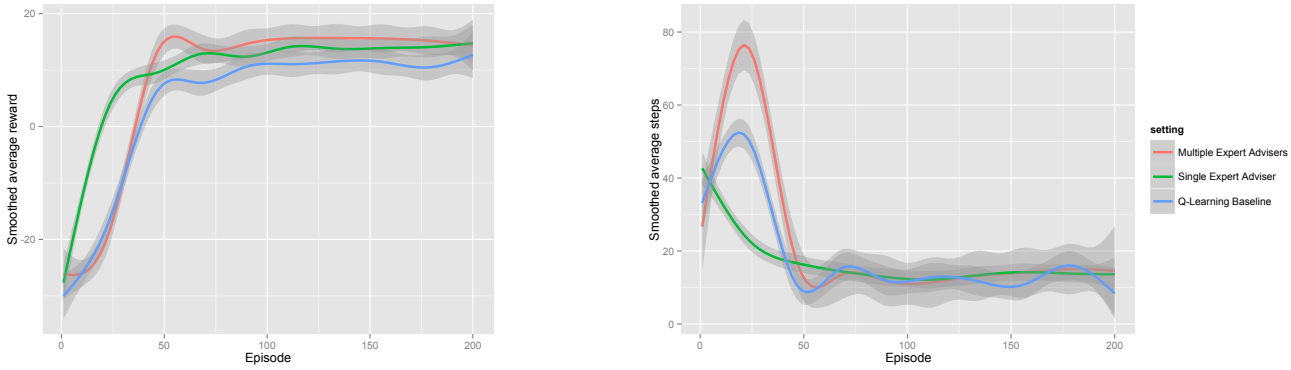
Figure 3 presents a first view over the results, offering a comparison between the group of experts, a single expert adviser and the Q-learning baseline in terms of rewards and steps. First, we notice that both cases involving human advice manage to surpass the baseline, with multiple experts advisers being asymptotically the best, by a slight difference. However, the single expert adviser does manage to converge faster, while reducing drastically the number of steps for the initial episodes.

Empirically, ensemble learning tends to generalize better in most cases, compared to an agent learning in isolation [**?**, **?**]. Nonetheless, for the multiple expert advisers, we can see that the gained cumulative reward stays close to the baseline. We can speculate that the lack of coordination might be detrimental to the overall performance. The reason for this might be that the tile-coded function approximator generalizes too much by assuming independent features. A single expert knows how to take advantage of this generalization in order to guide the agent downwards and then right. However, the variety in the advice from multiple humans makes coordinating this exploitation more difficult.

We take a closer look on how the function approximator affects our results. Figure 4 shows the normalized frequencies of advice given by a single adviser (a) and multiple expert advisers (b). We can see that the feedback is naturally less sparse for multiple advisers. The effect of the function approximator is depicted in sub-plots (c) and (d) for respectively a single and multiple advisers. As each feature is handled independently in the tile-coding, the advice gets generalized over all features separately. These plots show the extrapolation of advice over the x and y coordinates. The optimal policy is in short "first go down for a while, then go right" (w.r.t. the starting position of the agent, as shown in Figure 2). For a single adviser, this can easily be done by rewarding the right actions. This is demonstrated in sub-plot (c), where we have two lines where the effective advice is concentrated (i.e., one around $x = 500$ and one around $y = 400$). However, when all five advisers give a positive reward for going down, they all have to coordinate to go right afterwards at the same moment. This lack of coordination is shown in sub-plot (d), where the effective advice is more evenly distributed in certain portions of the state space. Notice that the single vertical line ($x = 500$) is still intense, meaning that all advisers agree upon going down from the starting position.

The decrease in performance can also be due to the simplicity of the policy to learn (i.e., go down, then right, as depicted in Figure 2b). The main difficulty of the problem is that it is easier for the agent to minimize its number of steps (see Figure 2a), rather than to learn a more complex combat behaviour. Such a sub-optimal strategy can easily be prevented using only one expert adviser, while having multiple advisers introduces unnecessary variance in the guidance [**?**].

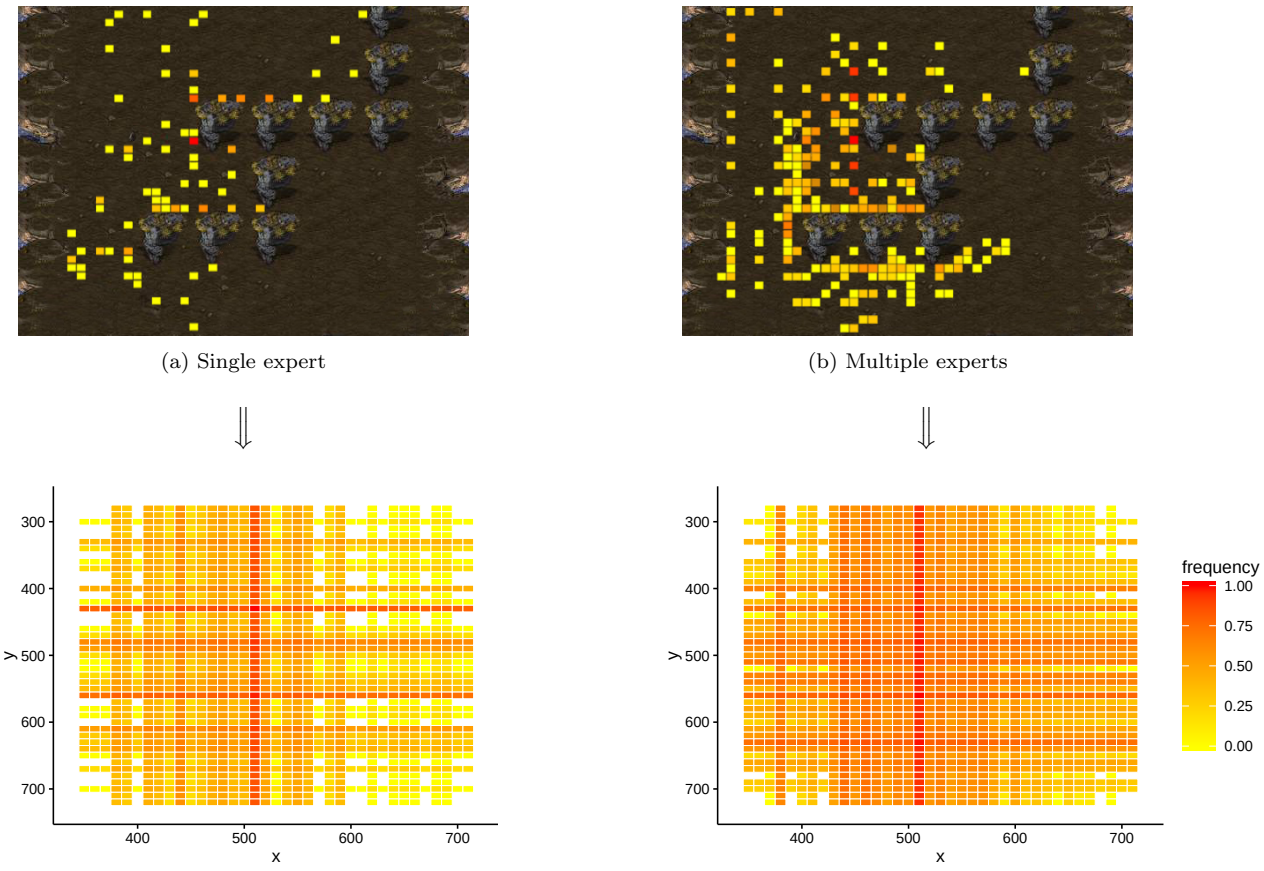
Thus, even though ensembles of advisers reduce the noise inherent to independent human advice, the function approximator generalizes the ensemble advice too much over single features, such that coordination to counter the extrapolated advice is necessary. On the other hand, when we have a



(a) Rewards

(b) Steps

Figure 3: Comparison of the average smoothed rewards and steps per episode over 9 trials between advice from multiple humans, advice from a single (expert) human, and a baseline without any human advice. The human advice is given in the first 5 episodes. The data is smoothed using local polynomial regression. The grey areas around the line plots represent the 95% confidence intervals associated with the smoothing.



(a) Single expert

(b) Multiple experts

(c) Single expert (through tile-coding)

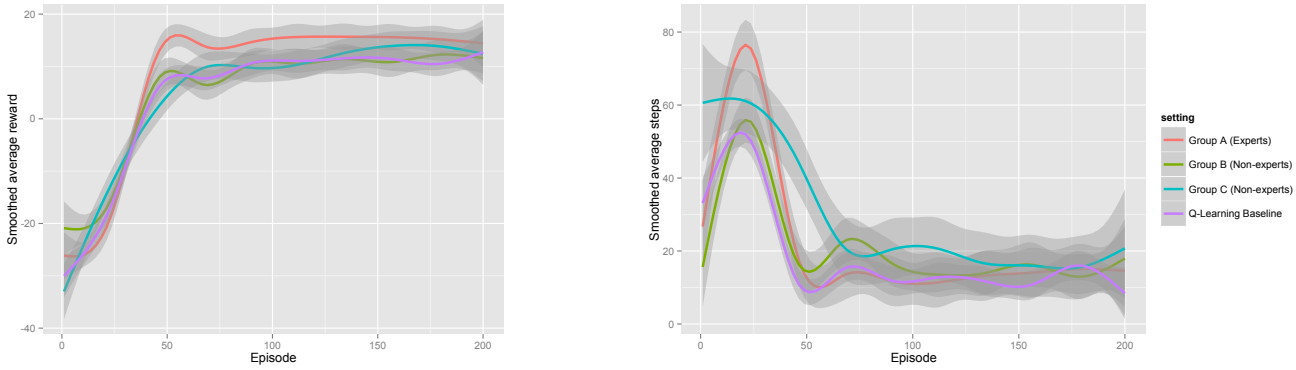
(d) Multiple experts (through tile-coding)

Figure 4: (a) and (b) show the normalized frequency of advice per visited state, given by respectively a single expert and group A, consisting of five expert advisers. (c) and (d) present the generalization of these frequencies over each feature independently when put through the tile-coder. (d) is thus a view of what the *effective* advice would be if all the feedback from the experts were accumulated and given to a single agent. These results show only the extrapolation over the x and y features.

simple target policy, the human advice could have less noise than the variance introduced by the ensemble learner.

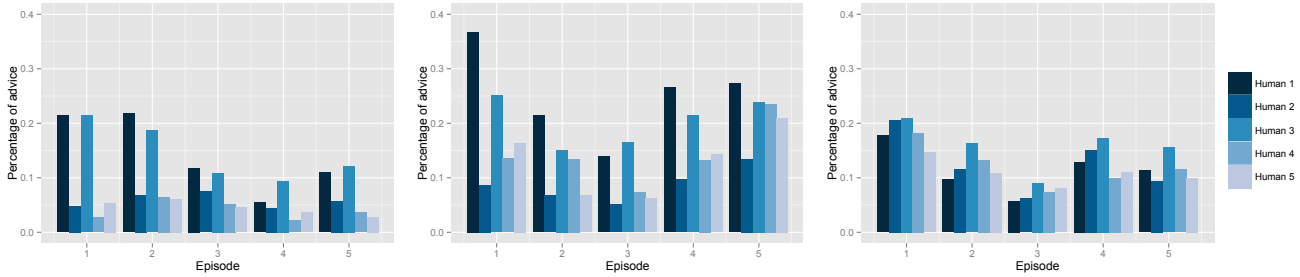
6.2 Experts vs Non-Experts

We now investigate the human advice over the different



(a) Rewards (b) Steps

Figure 5: Comparison of the average smoothed rewards and steps per episode over 9 trials between two non-expert groups and one expert group. The human advice is given in the first 5 episodes. The data is smoothed using local polynomial regression. The grey areas around the line plots represent the 95% confidence intervals associated with the smoothing.



(a) Group A (Experts) (b) Group B (Non-experts) (c) Group C (Non-experts)

Figure 6: Fraction of steps when advice was given by each human over the 9 trials per episode.

groups, and evaluate it in terms of impact on the convergence speed and overall performance. Figure 5 presents the comparison between the expert and the non-expert subject groups, against the Q-learning baseline in terms of rewards and steps. The expert group is the only one that manages to get a clear separation from the baseline, while group C requires the most episodes before convergence. In terms of steps, the groups generally follow the trend of the Q-learning case, although the expert group seems to have a slower start. Again, group C requires the most episodes before convergence. Moreover, we noticed during our experiments that the performance varies a lot from run to run, as we only allowed for positive feedback to be given and the advisers could not always contribute a lot to the agent's behaviour. Figure 6 presents the fraction of steps in which advice is given by each human in each group, for each of the first 5 episodes, over all the trials. Group C is the most homogeneous in terms of advice quantity, while for the other groups, there are one or two main contributors. We can link these findings back to the hypothesis that the lack of coordination worsens the convergence speed. The advice given by group A and B are mostly defined by one or two persons, which means these persons can coordinate the advice better. In contrast, the people in group C give an equal amount of advice, which makes coordination more difficult.

7. CONCLUSIONS

We introduced real-time human guided ensemble learning, a combination of ensemble learning with reward shaping that learns the advice from multiple experts on-line. We evaluated our approach in a StarCraft setting, controlling a single agent in combat against one enemy. We had three groups of five humans, one expert and two non-expert groups, giving feedback to our learner during the first episodes of a number of independent trials, and analysed the performance of the learning process in terms of convergence speed. We noticed that in our experimental setting, having multiple human advisers does not increase the performance of the agent. When compared to a single adviser, the learning process converges less quickly, and at about the same rate as without any human advice. Further analysis confirmed that multiple humans did indeed not succeed in outperforming a single expert adviser.

One of the things that could have affected our results, is the function approximator, as we noticed that it seemed to act in a way that was not suited for the relatively sparse human feedback. The fact that all feature dimensions are separately tiled, made the learner generalize too much over a specific action. We noticed that after encouraging a certain action a few times, the way our function approximator worked caused these actions to be rewarded in other unrelated regions of the state space. Even though a single expert could still compensate for this flaw, multiple experts cannot avoid it, due to a lack of coordination.

Additionally, the optimal strategy might be too simple and transparent to the human advisers to have an ensemble of agents, in contrast to a single human expert, whose advice function is close to the actual optimal strategy the agent has to learn.

Though we have not been able to irrefutably conclude that combined human advice provides an advantage, the results of our experiments indicate that further research can be beneficial in order to obtain a real-time crowd-sourcing framework for complex RL settings. In future work, we will adapt the function approximator to avoid generalization over independent features and re-evaluate our approach. Additionally, we will look into scenarios for which a more complex strategy should be adapted in order to win the game.