LIEF: Learning to Influence through Evaluative Feedback
Ramona Merhej, Mohamed Chetouani

To cite this version:
Ramona Merhej, Mohamed Chetouani. LIEF: Learning to Influence through Evaluative Feedback. Adaptive and Learning Agents Workshop (AAMAS 2021), May 2021, London (virtual), United Kingdom. hal-03224579

HAL Id: hal-03224579
https://hal.archives-ouvertes.fr/hal-03224579
Submitted on 11 May 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
We present a multi-agent reinforcement learning framework where the humans, not the environment itself. It is particularly seen are widely used. This additional reward function is generated by defined by the environment only. Nonetheless and again, in all of the above, this type of feedback speed up learning [4, 10, 21], enhance coordination [13, 21] etc. Communication can take several forms. For example, agents may communicate by sending messages [10], sharing intentions [13] or experiences [4], advising actions [21] to one another etc. Inte- 

Communicating through action selection instead of feedback can be more form of a positive or negative reward is attributed. This feedback, defined by a reward function, "is the most succinct, robust, and transferable definition of the task" [17]. In Multi-Agent Reinforce-
ment Learning (MARL), several agents interact with each other and with the environment to solve a common or disparate tasks (i.e., they have a common or disparate reward functions). In the general case, defined as a Markov Game, both the reward function of an agent as well as the transition function of the environment, can have explicit dependencies on the joint action of all agents. The expected return of an agent depends on the policy it follows as well as the policies of all its peers [19]. We note however that, although agents influence the performance of one another, they do not alter the structure of the reward function itself.

Additionally, in the multi-agent setting, other more direct inter-action mechanisms can be present between agents. Many MARL algorithms integrate and allow communication between the agents. Communication can take several forms. For example, agents may communicate by sending messages [10], sharing intentions [13] or experiences [4], advising actions [21] to one another etc. Integrating such feedback has shown to improve performance [4, 10], speed up learning [4, 10, 21], enhance coordination [13, 21] etc. Nonetheless and again, in all of the above, this type of feedback does not alter other agents' task or reward function which remains defined by the environment only.

However, in real-life situations, punishments and remunerations are widely used. This additional reward function is generated by the humans and not the environment itself. It is particularly seen when the environment presents conflicting goals to its agents or incentives for exploitation. Consider as an example the taxing system, the healthcare system or other public goods. In all of these examples, the systems themselves are unstable and entice exploitation. Fines and sanctions are put in place by the state to modify the dynamics and make exploitation and fraud less desirable and encourage and stabilise socially beneficial behaviours. These measures incite people to diverge from what would have otherwise been their optimal strategy. Remunerations are also a mean of influencing others' behaviours. For instance, when the number of users sending funds over the bitcoin blockchain exceeds the number of transactions that can be processed, users encourage miners to validate their transaction by remunerating them with higher transaction fees. Likewise, some business owners use cash, store credit, discounts etc. to encourage their customers to recommend them to other friends. Remunerations also appear in personal relations, education or other similar interactions. Despite their effectiveness, both sanctions and remunerations are costly to implement and the benefits extracted from them need to outweigh their costs.

While cooperative behaviours in an opponent can sometimes be encouraged without explicit evaluative feedback, e.g., using the Tit-for-Tat strategy [1] for the iterated prisoner dilemma, such a retaliating strategy is not available or optimal for all games. Consider Public Good Games where agents contribute to a common pool and receive in return a shared reward proportional to the collected pool. Punishing a non-contributing agent by not contributing oneself results in punishment for everyone instead of a targeted one. Furthermore, in the bitcoin blockchain example, no retaliation policy exists for the user that encourages miners to validate his transaction. Additionally, in real-life situations, punishing or rewarding through action selection instead of feedback can be more challenging to implement. For instance, boycotting a company for animal testing is more challenging to implement than fining such policies (e.g., when the company holds a near-monopoly on the market). Evaluative feedback is a universal form of punishment and remuneration and has the advantage of being simultaneously easy to target and simple to implement.

In our work, we present a Multi-Agent Reinforcement Learning framework where agents learn to influence each other, not by action selection or communication, but directly using costly remunerations or punishments, that we denote by evaluative feedback. The framework is useful when agents have conflicting goals. Rewarding each other is a way of sharing preferences about the opponent’s behaviours. Through mutual sharing of these preferences, we hope that agents find arrangements and compromises that allow mutual co-existence instead of mutual destruction. The new arrangements emerge from the reshaped goals (i.e., the reshaped reward functions) of every agent by its opponent. We propose Learning to Influence through Evaluative Feedback (LIEF), an algorithm designed to learn
The influence that agents can exercise on each other has been taken 
into account in some MARL algorithms where agents adapt their 
choices with respect to their opponent. Notably, the learning algo-
rithm of an opponent is sometimes integrated in one’s own learning 
algorithm. One suggested method is gradient ascent with policy 
prediction [25]. Here the strategy of the opponent is forecasted 
based on the current policy parameters and the gradient is com-
puted taking this change into account. Learning with Opponent 
Learning Awareness (LOLA) [8] on the other hand, differentiates 
through the variations of the opponent to actively shape their learn-
ning and consequently, manages to reach cooperative equilibria in 
some multi-agent settings. Stable Opponent Shaping (SOS) [14] 
incorporates both policy prediction and opponent shaping which 
increases stability while simultaneously escaping saddle points.

While the above mentioned algorithms, try to influence a gradient-
based learning opponent, Learning and Influencing Latent Intent 
(LILA) [24] influences a handcrafted opponent that can select for 
every episode to implement one of several pre-defined policies. The 
set of pre-defined policies is small and comprises 2-3 policies. 
The selection of the policy by the opponent at the beginning of 
an episode is not random but depends on the previous interaction. 
LILA models and predicts the opponent’s next policy choice and en-
sures that the current interaction results in a favourable opponent 
policy in the next episode.

2 RELATED WORK

The influence that agents can exercise on each other has been taken 
into account in some MARL algorithms where agents adapt their 
choices with respect to their opponent. Notably, the learning algo-
rithm of an opponent is sometimes integrated in one’s own learning 
algorithm. One suggested method is gradient ascent with policy 
prediction [25]. Here the strategy of the opponent is forecasted 
based on the current policy parameters and the gradient is com-
puted taking this change into account. Learning with Opponent 
Learning Awareness (LOLA) [8] on the other hand, differentiates 
through the variations of the opponent to actively shape their learn-
ning and consequently, manages to reach cooperative equilibria in 
some multi-agent settings. Stable Opponent Shaping (SOS) [14] 
incorporates both policy prediction and opponent shaping which 
increases stability while simultaneously escaping saddle points.

While the above mentioned algorithms, try to influence a gradient-
based learning opponent, Learning and Influencing Latent Intent 
(LILA) [24] influences a handcrafted opponent that can select for 
every episode to implement one of several pre-defined policies. The 
set of pre-defined policies is small and comprises 2-3 policies. 
The selection of the policy by the opponent at the beginning of 
an episode is not random but depends on the previous interaction. 
LILA models and predicts the opponent’s next policy choice and en-
sures that the current interaction results in a favourable opponent 
policy in the next episode.

In the context of influence for long term future benefits, Prognos-
ticator [2] is an algorithm that allows an agent in a non-stationary 
environment to forecast future performances and hence select to 
minimise performance in some episodes if that results in a future 
increase in performance. Here the future changes neither result 
from a learning opponent, nor from a conditionally changing one 
but from a smoothly varying environment.

All these enumerated works try to influence the dynamics of an 
opponent or an environment by selecting influential actions. 
On the other hand, in our work, we aim to influence an opponent’s 
behaviour using directly rewards instead of regular actions.

Recovering or optimising a reward structure is at the center of 
interest of many RL problems. Inverse Reinforcement Learning (IRL) 
[18] for example, tries to recover the reward function that could 
be a priori for a given optimal policy or set of trajectories. AutoRL 
and Evolution Strategies (ES) have also been used to explore and 
evolve different reward functions with the goal of finding the one 
that facilitates the learning of a predefined task [3, 7]. Moreover, in 
some adversarial RL problems [22, 26], an attacker aims to poison

a learner’s reward function with the goal of enforcing a predefined 
target policy on that learner.

Yet several differences are notable between these works and ours. 
First, while we propose a reinforcement learning method to learn 
how to reward an opponent, the problem in the above examples 
is usually defined as a control problem and standard optimisation 
techniques are used to extract the optimal solution. Second, while 
the target policy is predefined for adversarial attacks and IRL, it is 
never explicitly given in our framework.

To the best of our knowledge, the only other work that introduces 
inter-agent evaluative feedback is a paper that studies the concept 
however, using the classical reinforcement learning objective of 
maximising episodic returns do not learn how to effectively gift 
their peers and stop using this action when gifting is costly.

3 DEFINITIONS AND NOTATIONS

Our work concerns general n-player stochastic games defined by a 
tuple \((S, \mathcal{A}, P, r, \mu_0)\). Here, \(S\) is the set of states, \(\mu_0\) the initial 
state distribution and \(\mathcal{A}\) the Cartesian product of the sets of 
actions of all individual players. At every timestep, each agent 
selects from his set of actions an action \(a_i\) yielding a joint action \(a = 
\{a_1, \ldots, a_n\}\). The system in state \(s\), then transitions to \(s'\) following 
the transition probability function \(P(s'|s, a) : S \times \mathcal{A} \times S \rightarrow [0, 1]\). 
Finally, the reward function \(r\) representative of the underlying tasks, 
evaluates and distributes for every player \(i\), a reward according to 
\(r_i(s, a, s') : S \times \mathcal{A} \times S \rightarrow \mathbb{R}\).

We extend this setting to allow for inter-agent evaluative feed-
back. In this model, agents do not only receive rewards from their 
environment, but also directly from other peers. Therefore, we 
enow endow every agent with an additional set of evaluative actions 
\(\mathcal{U}_i\) and we denote by \(\mathcal{U}\), the Cartesian product of these sets \(\mathcal{U} = 
\prod_{i=1}^{n} \mathcal{U}_i\). We suppose that such evaluative actions can be costly. 
Every agent \(i\), now incurs a cost based on his selected action \(u_i\) 
and receives evaluative feedback based on his peers’ selected ac-
tions \(u_{-i}\). The costs are determined by a cost function \(c\) such that 
\(c_i(s, u_i, s') : S \times \mathcal{U}_i \times S \rightarrow \mathbb{R}\) and the peers’ feedback is 
calculated by the feedback function \(f\) for where for every agent \(i\) we have 
\(f_i(s, a_{-i}, s') : S \times \mathcal{U}_{-i} \times S \rightarrow \mathbb{R}\).

The original stochastic game represented by the tuple \((S, \mathcal{A}, P, r, \mu_0)\) 
is now extended with an additional evaluative action set \(\mathcal{U}\) and two 
reward functions \(c\) and \(f\) representing respectively the cost of evalu-
ating others and the additional feedback received. The resulting 
dynamics are now described by the tuple \((S, \mathcal{A}, \mathcal{U}, P, r, c, f, \mu_0)\).

4 METHODS

To solve the described problem, we equip every agent with two de-
coupled policies with different objectives: the classical game policy 
and the feedback policy. The task or game policy of agent \(i\), denoted 
by \(\pi^g_i\), maps into the action space \(\mathcal{A}_i\) while the feedback policy \(\pi^f_i\) 
maps into the action space \(\mathcal{U}_i\). The objective of the game policy is a 
classical RL objective i.e., solving a task by maximising the expected 
return within an episode. Note however, that the task, modelled by 
a reward function, is not only defined by the environment, but also 
generated by the agent’s peers. The original task defined by \(r\) is re-
shaped into the function \(r + f\) from the tuple \((S, \mathcal{A}, \mathcal{U}, P, r, c, f, \mu_0)\).
The objective of the feedback policy is of a different nature. While the game policy is concerned with maximising the returns within an episode, the feedback policy tries to influence its opponent to make future interactions more beneficial. We note that in a multi-agent setting, the returns of one agent depend on all other agents’ actions. Therefore, the performance of a fixed agent can vary if its opponent changes policies. We consider a future interaction to be more beneficial for an agent, if keeping his own game policy fixed, the opponent’s game policy update results in higher payoffs for said agent. The algorithm we propose for this task is called Learning to Influence through Evaluative Feedback (LIEF). Figure 1 depicts the new extended game dynamics.

4.1 The Game Policy

In classical RL, an agent searches for an optimal policy to solve a task assigned to it by its environment and defined by a reward function. The game policy in our framework is analogous to the classical RL policy. It assumes a stationary world and maximises the expected episodic return. However, in our framework, feedback is received simultaneously from the environment and other agents. The expected episodic return, defined by $r + f$, is conditioned on all agents’ game policies $\pi^i$ (that we parametrise with $\theta^i$) and on the opponents’ feedback policies $\pi^f_{-i}$ (parametrised with $\theta^f$).

Given an episode horizon $T$, agent $i$’s game policy objective function is defined as

$$J_i^f (\theta_i^f, \theta_{-i}^f, \theta_{-i}^f) = \mathbb{E}_{\theta_i^f, \theta_{-i}^f} \left[ \sum_{t=0}^{T-1} y^r_t r_{i,t} + f_{i,t} \right]$$

where $y^r$ is a discount factor.

Parameters $\theta_i^f$ are updated using the update rule

$$\theta_i^{f,k+1} = \theta_i^{f,k} + \eta_f \nabla \theta_i^f J_i^f (\theta_i^f, \theta_{-i}^f, \theta_{-i}^f)$$

where $\eta_f$ is the learning rate for updating the game policy and $k$ the optimisation step.

4.2 The Feedback Policy

On the other hand, the goal of the rewarding or feedback policy is to shape the objective of the opponent to make it more compatible with its own. To achieve this target, the feedback network tries to maximise the variations in returns resulting from one opponent optimisation step. The idea is detailed in Figure 2 and the associated objective function of the feedback policy is given by

$$J_i^f (\theta_i^f, \theta^f) = \mathbb{E}_{\theta_i^f, \theta^f} \left[ \sum_{t=0}^{T-1} y^r_t r_{i,t} + f_{i,t} \right]$$

$$(\Delta \theta^f) = \nabla \theta_i^f \mathbb{E}_{\theta_i^f, \theta^f} \left[ \sum_{t=0}^{T-1} y^r_t r_{i,t} + f_{i,t} \right]$$

which yields the final objective function

$$J_i^f (\theta_i^f, \theta^f) = (\Delta \theta^f)^T \nabla \theta_i^f \mathbb{E}_{\theta_i^f, \theta^f} \left[ \sum_{t=0}^{T-1} y^r_t r_{i,t} + f_{i,t} \right]$$

We can now use a gradient ascent update rule

$$\theta_i^{f,k+1} = \theta_i^{f,k} + \eta_f \nabla \theta_i^f J_i^f (\theta_i^f, \theta^f)$$

where $\eta_f$ is the learning rate for updating the feedback policy and $k$ the optimisation step.

We need now to evaluate the term $\nabla \theta_i^f J_i^f (\theta_i^f, \theta^f)$. To simplify notations, we use $R_i$ to refer to $\mathbb{E}_{\theta_i^f, \theta^f} \left[ \sum_{t=0}^{T-1} y^r_t r_{i,t} \right]$ and $C_i$ to refer to $\mathbb{E}_{\theta_i^f, \theta^f} \left[ \sum_{t=0}^{T-1} y^c_t c_{i,t} \right]$. The product rule gives us...
We propose to test LIEF in two different scenarios. We begin with the feedback and game policy are trained alternately for a number of episodes. All actors or critics are linear functions of the state dynamics and stabilise the more beneficial and cooperative point.

In all experiments, we train the feedback and game policy alternately for \( K_f \) and \( K_g \) update-steps respectively. The equivalent DiCE objective \([9]\) of the feedback policy objective is constructed and the corresponding loss is used during training to mediate the errors in estimating second order derivatives of a surrogate loss.

The game policy can be trained using an actor-critic architecture and the feedback policy with the REINFORCE algorithm.

A pseudo-code is given in Algorithm 1.

### 5 EXPERIMENTS AND RESULTS

We propose to test LIEF in two different scenarios. We begin with an environment called Teacher-Student. Here, the student agent receives no rewards from the environment and its only feedback comes from the teacher. We make sure that, without the teacher, the learning curve of the student is flat. Learning is purely directed by the teacher agent which has to learn how to use evaluative feedback to bring the student to accomplish a task.

In the second case, we test our framework on the iterated prisoner’s dilemma. In this game, mutual cooperation is more beneficial than mutual defection. However, with the environment rewards only, the point of mutual cooperation is unstable and naive learners eventually converge to a less optimal but stable defective point.

We test if, with inter-agent feedback, agents can modify the game dynamics and stabilise the more beneficial and cooperative point.

#### 5.1 Hyper-parameters

In all experiments, we train the feedback and game policy alternately for \( K_f = 35 \) and \( K_g = 5 \) update-steps, using a batch size of 4096 episodes. All actors or critics are linear functions of the state, and all states are one-hot encoded vectors. The learning rates of both actor and critic of the game policy are set to \( \eta_f = 1 \) and we use a discount factor \( \gamma_f = 0.8 \). The learning rate of the feedback policy is set to \( \eta_g = 0.1 \) and the discount factors are \( \gamma_r = 0.99 \) and \( \gamma_c = 0.9 \).

#### 5.2 Teacher-Student Environment

We begin by testing LIEF in a simple Teacher-Student chain environment depicted in Figure 3. A student, physically present in the environment, can at every timestep, choose to either move left or right. However, the environment is uninteresting to the student and doesn’t provide him any rewards (\( r = 0 \) for \( s \in S \)). The teacher on the other hand, would like the student to reach the rightmost cell in the chain since that state yields a positive reward of \( r_t = +1 \) to the teacher. The goals of the agents are not at conflict but nevertheless, uncorrelated. The teacher has to motivate the student somehow to move right since without extrinsic intervention, actions right and left would be equally desirable for the student.

The environment is a representation of situations where one individual has an interest in accomplishing a task but lacks the skills to do so. Another agent, with the necessary skills, has no personal interest in getting the job done. Depending on the relationship between the two individuals, the former needs to either motivate the latter by paying him a remuneration for getting the job done or punishing him for not getting the task done.

In our experiment, we provide the teacher with a binary punitive action set \( \mathcal{U} = \{0, 1\} \). Both the teacher and the student observe the state of the environment (i.e., the chain position of the student.

\[
\begin{align*}
\nabla_{\theta_f} J_f (\theta_f^t; \theta_g) &= \nabla_{\theta_f} \Delta \theta_g^t (R_t + C_t) \\
&= \left( \nabla_{\theta_f} \Delta \theta_g^t (R_t + C_t) \right)^T \nabla_{\theta_g}^t (R_t + C_t)
\end{align*}
\]

Using a similar derivation as in LOLA \([8]\), that we do not detail here, we have

\[
\Delta \theta_g^t = \eta_g \nabla_{\theta_g} E_j \left[ \sum_{t=0}^{T} \gamma_j^t (r_{j,t} + f_{j,t}) \right]
\]

\[
\Delta \theta_f^t = \eta_f \nabla_{\theta_f} E_i \left[ \sum_{t=0}^{T} \gamma_i^t (r_{i,t} + f_{i,t}) \right]
\]

\[
\nabla \theta_g^t = \nabla_{\theta_g} \Delta \theta_g^t = \nabla_{\theta_g} \nabla_{\theta_g}^t (R_t + C_t)
\]

\[
\nabla \theta_f^t = \nabla_{\theta_f} \Delta \theta_f^t = \nabla_{\theta_f} \nabla_{\theta_f}^t (R_t + C_t)
\]

\[
\nabla \theta_g^t = \nabla_{\theta_g} \Delta \theta_g^t = \nabla_{\theta_g} \nabla_{\theta_g}^t (R_t + C_t)
\]

\[
\nabla \theta_f^t = \nabla_{\theta_f} \Delta \theta_f^t = \nabla_{\theta_f} \nabla_{\theta_f}^t (R_t + C_t)
\]

\[
\nabla \theta_g^t = \nabla_{\theta_g} \Delta \theta_g^t = \nabla_{\theta_g} \nabla_{\theta_g}^t (R_t + C_t)
\]

\[
\nabla \theta_f^t = \nabla_{\theta_f} \Delta \theta_f^t = \nabla_{\theta_f} \nabla_{\theta_f}^t (R_t + C_t)
\]

\[
\nabla \theta_g^t = \nabla_{\theta_g} \Delta \theta_g^t = \nabla_{\theta_g} \nabla_{\theta_g}^t (R_t + C_t)
\]

\[
\nabla \theta_f^t = \nabla_{\theta_f} \Delta \theta_f^t = \nabla_{\theta_f} \nabla_{\theta_f}^t (R_t + C_t)
\]

\[
\nabla \theta_g^t = \nabla_{\theta_g} \Delta \theta_g^t = \nabla_{\theta_g} \nabla_{\theta_g}^t (R_t + C_t)
\]

\[
\nabla \theta_f^t = \nabla_{\theta_f} \Delta \theta_f^t = \nabla_{\theta_f} \nabla_{\theta_f}^t (R_t + C_t)
\]
and whether the student selected action right or action left in the previous timestep). The student can then select to either move right or left while the teacher can select to either punish the student or not. Not punishing, or selecting $u = 0$, returns no feedback to the student ($f = 0$) and incurs no cost on the teacher ($c = 0$). However, choosing to punish, i.e., $u = 1$, returns a negative feedback to the student ($f = -1$) and is equally costly to the teacher ($c = -1$). Figure 4 illustrates the interactions and feedback flow in this setting.

We set a fixed timestep of $T_e = 4$ per episode and run our experiment with 10 different seeds. From the results plotted in Figure 5, we can see that during feedback policy training, the teacher learns to punish the student when the latter chooses the non beneficial action of going Left (green line rises during the 35 feedback policy update-steps). Consequently, during student learning, the feedback policy of the teacher causes the student to increase the frequency with which he chooses to go right (blue line rises). Nevertheless, some shortcomings are visible in the algorithm. The teacher has trouble completely cancelling punishments of actions in his favour (the red line doesn’t decrease to zero). Deeper examination of this cause and an improvement of the algorithm are needed. When training the student without any feedback from the teacher, the preference of the student for action Right or Left remains constant throughout training (all rewards are zero and hence all gradients are zero). The student’s preference for action Right in Figure 5 is solely constructed by the teacher.

We note that our definition of Teacher-Student differs from that commonly seen in MARL [5, 6, 11, 12, 20, 21] where both the teacher and the student interact with the environment. There, the rewards or feedback come exclusively from the environment. The teacher, generally more skilled, is more of a guide to help the student achieve a task defined by the environment. In our case, the teacher is not more skilled than the student but lacks the means to perform a task he’d like accomplished. He cannot, like in the classical Teacher-Student case, give any demonstrations or examples to guide the student. His goal is to construct a target goal for the student (through evaluative feedback) that is compatible with his own goal.

### 5.3 Iterated Prisoner Dilemma

In a second experiment, we propose to test the efficacy of LIEF in an antagonistic environment. We select for this purpose the iterated prisoner’s dilemma with returns shown in Table 1.

<table>
<thead>
<tr>
<th>Actions</th>
<th>A2 - C</th>
<th>A2 - D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 - C</td>
<td>(-1, -1)</td>
<td>(-3, 0)</td>
</tr>
<tr>
<td>A1 - D</td>
<td>(0, -3)</td>
<td>(-2, -2)</td>
</tr>
</tbody>
</table>

In the one-shot version of the game, the environment raises in every player a preference to defect no matter the opponent’s strategy. In fact, for a defecting opponent, defection results in one point more than cooperation (-2 compared to -3). Similarly, for a cooperative opponent, defection results in one point more than cooperation (0 compared to -1). As a result, both players learn to defect and converge to the bottom right cell of the table with an average of -2 points per player. We note here that, both players can in fact increase their returns by switching simultaneously to cooperation. However, this point, although more beneficial, is unstable.

In the iterated version of the game, which is the one we adopt for our experiment, agents play the game repeatedly against one
We present a multi-agent reinforcement learning framework with 10 different seeds. The results are plotted in Figure 7.

While cooperation is usually difficult to sustain in the IPD, Figure 7 shows that agents converge to cooperative behaviours. We can see that during the training of the feedback policy, the green lines, indicating the rate with which agents punish a defective behaviour, increase while the red lines, indicating the rate at which they cooperate increase. As a result, during game policy training, cooperative actions become more advantageous than defective ones and players converge to total cooperation.

**DISCUSSION AND CONCLUSION**

We present a multi-agent reinforcement learning framework extended to incorporate inter-agent evaluative feedback. Leveraging the fact that the reward function in RL is the fundamental definition of a task, we allow agents, through this framework, to construct or modify the tasks of their peers.

In the teacher-student case (see Section 5.2), the teacher, using evaluative feedback, was constructed a target goal for the student opponent. They are endowed with a memory and observe at every timestep the actions taken by themselves and their opponent in the previous timestep. Accessing this info, allows some strategies to converge to cooperative behaviours such as the Tit-for-Tat strategy. In Tit-for-Tat, an agent punishes at timestep t, an opponent that defected at the previous timestep t − 1. The punishment is implemented as a retaliation and the agent reciprocates an opponent’s defection at t − 1 with a defection at t. Although effective, Tit-for-Tat is not trivial to be found and naive reinforcement learners generally converge to defective behaviours.

By introducing inter-agent feedback, agents can modify the reward table and hence change the game dynamics. Instead of retaliation, agents may use punishment to compel cooperative behaviours from their opponents. If every agent manages to make action defect less desirable to their rival (e.g., by punishing this action), they can converge to cooperative behaviours without the risk of being exploited. Figure 6 shows the flow of the added inter-agent feedback and the resulting costs on the agents. In our case, we use a binary punitive action set $U_i = \{0, 1\}$ for each agent and the resulting feedback and costs are such that $f_i(u_{-i}) = -3u_{-i}$ and $c_i(u_i) = -1u_i$.

**ACKNOWLEDGMENTS**

This work has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement No 76595.

**REFERENCES**


