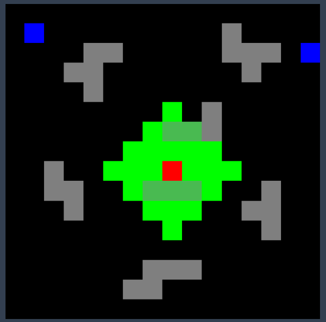


Exploring the Impact of Tunable Agents in Sequential Social Dilemmas

David O’Callaghan
ocallaghan1@gmail.com

Patrick Mannion
patrick.mannion@nuigalway.ie

School of Computer Science

National University of Ireland Galway

Contents

OÉ Gaillimh
NUI Galway

1. Abstract
2. Background
3. Related Work
4. Methodology
5. Experimentation
6. Results
7. Future Work

Abstract

- Reinforcement learning agents are most commonly trained to learn some fixed behaviour.
- However, if a different form of agent behaviour is required, the agent typically needs to be retrained.
- In this study, we use a method to design agents whose behaviours can be tuned after training using techniques from multi-objective reinforcement learning.
- We empirically show that the tunable agents framework enables easy adaption between cooperative and competitive behaviours in sequential social dilemmas.

Background

Multi-objective reinforcement learning

- Standard RL methods solve a sequential decision-making problem by optimising a single objective.
- Many real-world problems are multi-objective by nature and these objectives can be in conflict with each other.
- The MDP is extended to a multi-objective MDP, where the agent receives vectorized rewards.

$$\mathbf{V}^{\pi}(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^{k} \mathbf{r}_{t+k+1} \middle| \pi, s_t = s\right]$$

Background

Multi-objective reinforcement learning continued…

- One approach to solving MOMDPs is to use a scalarisation function
- General form:

$$V^{\pi}_{\mathbf{w}}(s) = f(\mathbf{V}^{\pi}(s), \mathbf{w})$$

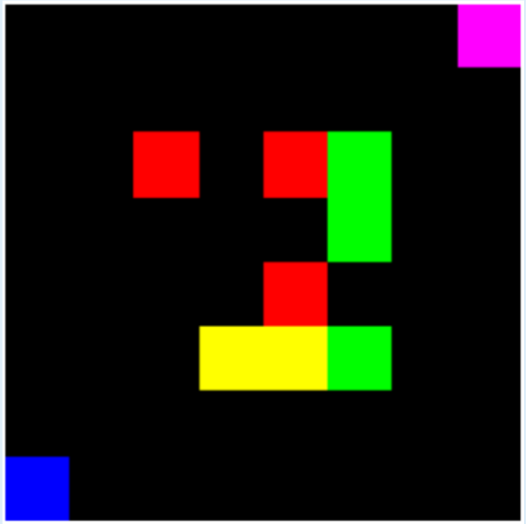
- Linear scalarisation:

$$V^{\pi}_{\mathbf{w}}(s) = \mathbf{w} \cdot \mathbf{V}^{\pi}(s)$$

- If $\mathbf{w}$ is unknown in the learning stage, a set of policies can be learned using a range of values for $\mathbf{w}$.

Related Work

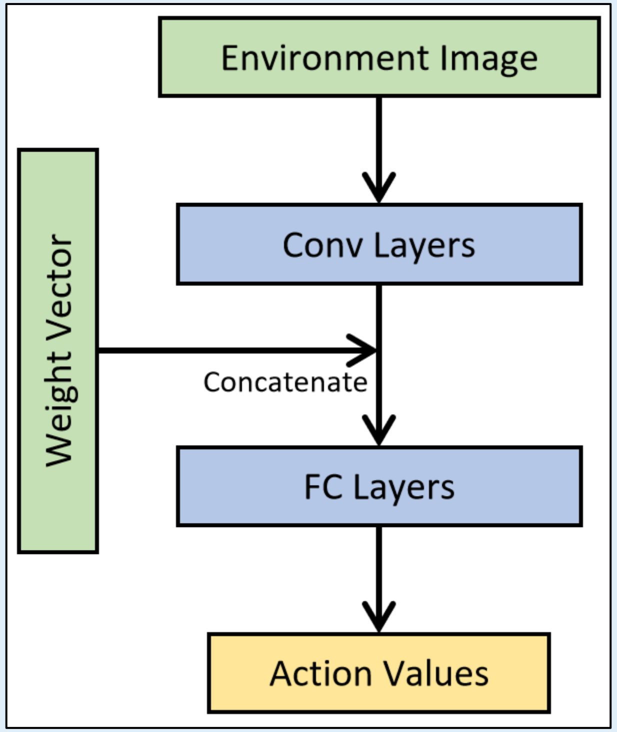
- Källström and Heintz (2019) introduced an agent architecture for training agents that can prioritize a set of objectives in a multi-objective environment during runtime.
- The study showed that agents trained using this architecture could approximate policies of agents trained with fixed preferences over objectives.
- Used 2 gridworld environments with a single trainable agent in experiments.



Källström, J., & Heintz, F. (2019). Tunable dynamics in agent-based simulation using multi-objective reinforcement learning. In *Adaptive and Learning Agents Workshop (ALA-19) at AAMAS, Montreal, Canada, May 13-14, 2019* (pp. 1-7).

Methodology

- The framework used for training tunable agents involves combining linear scalarization of rewards with the DQN algorithm.
- The neural network is conditioned on both the environment state and weight vector.
- The weight vector is re-sampled repeatedly during training.



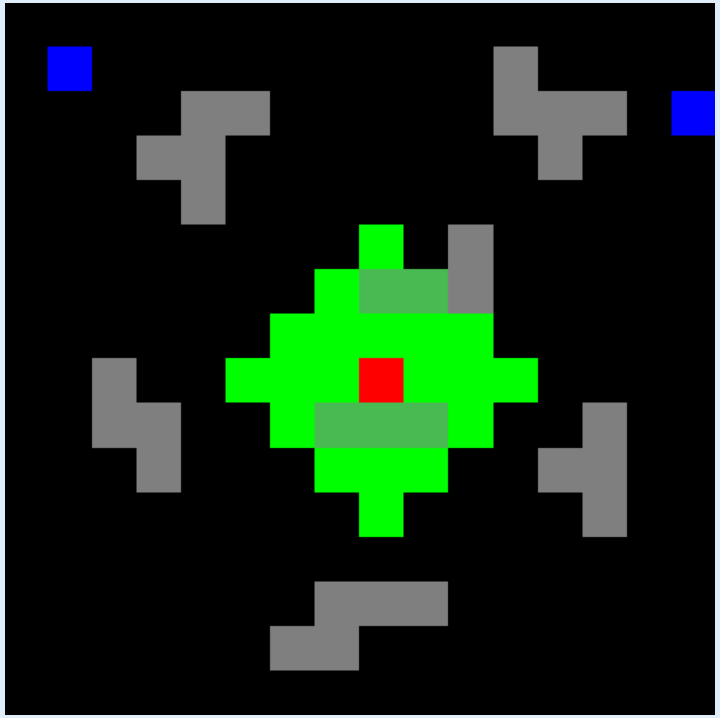
Methodology

The tunable DQN agent algorithm used in our work. *Adapted from Källström and Heintz (2019) for multi-agent settings.*

Initialise ϵ to 1.0
Set T to the number of episodes to start training after
for *episode* $\leftarrow 1$ **to** M **do**
    Perform each command individually for agent $i = 1, \ldots, n$ where n is the number of agents
    Get the initial state s_i from the environment
    Sample a set for preference weight vector ($\mathbf{w}_i$) from the preference weight sample space
    while s_i **is not** *terminal* **do**
        Choose action a_i from state s using ϵ-greedy policy: a_i = `agent[`i`].get_action(`$s_i, \mathbf{w}_i, \epsilon$`)`
        Take action a_i and observe reward vector $\mathbf{r}_i$ and next state s'_i
        Store experience in replay memory: `memory[`i`].append(`$s_i, a_i, \mathbf{r}_i, s'_i, \mathbf{w}_i$`)`
        $s_i \leftarrow s'_i$
    end
    Decay ϵ
    if *episode* $> T$ **then**
        minibatch $\leftarrow$ `memory[`i`].sample()`
        `agent[`i`].train(`*minibatch*`)`
    end
end

Experimentation

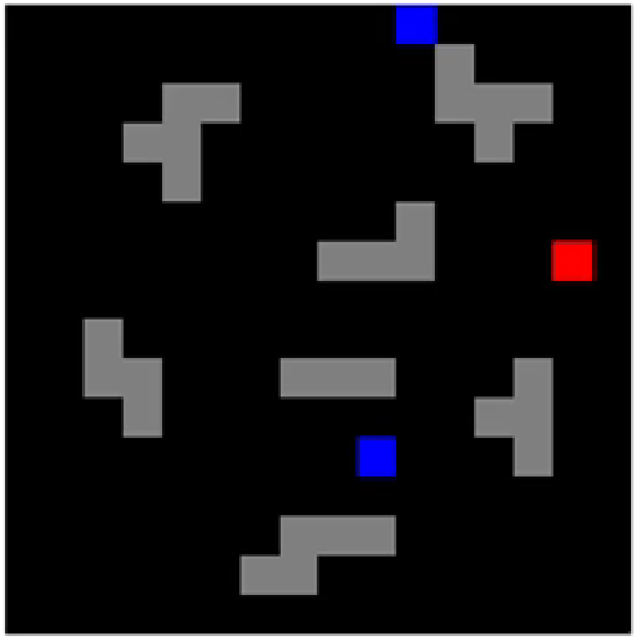
- In this study, we applied the tunable multi-agent framework to a multi-objective version of the Wolfpack environment from Leibo et al. (2017).
- 16x16 pixel-based gridworld. Multiple predator agents (blue) try to capture a prey (red).
- Team-capture if a predator is within the capture-radius (green area) when the other predator captures the prey. Lone-capture otherwise.
- Team-captures (cooperative) and lone-captures (competitive) are rewarded as separate objectives.



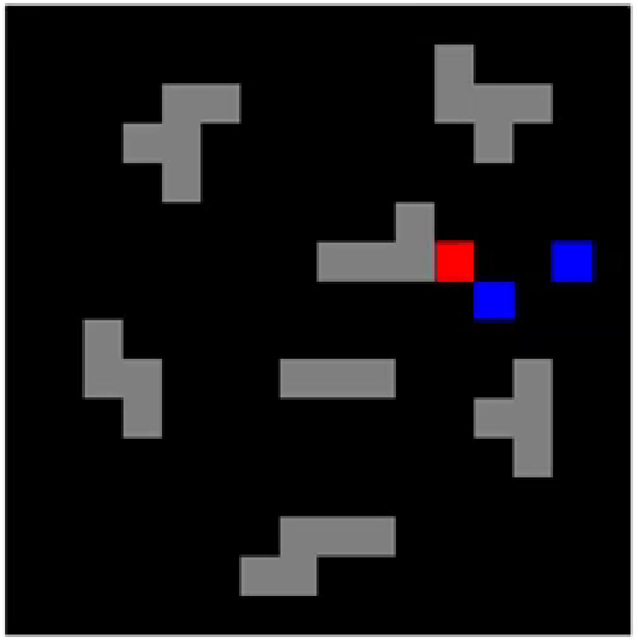
Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., & Graepel, T. (2017, May). Multi-agent Reinforcement Learning in Sequential Social Dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (pp. 464-473).

Results

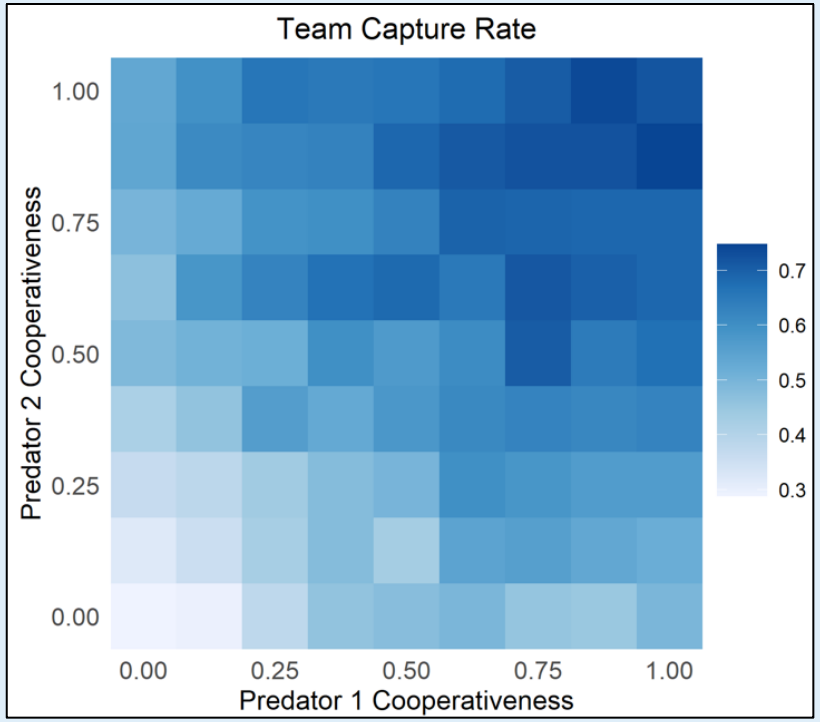
Competitive

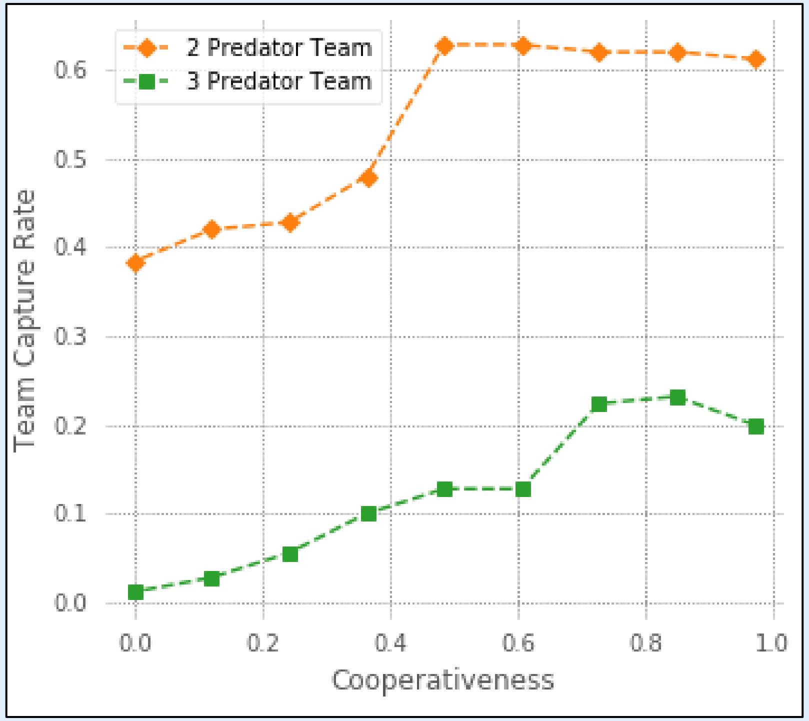


Cooperative



Results


Team Capture Rate
1.00
0.75
0.50
0.25
0.00
Predator 2 Cooperativeness
0.00
0.25
0.50
0.75
1.00
Predator 1 Cooperativeness
0.7
0.6
0.5
0.4
0.3


2 Predator Team
3 Predator Team
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Team Capture Rate
0.0
0.2
0.4
0.6
0.8
1.0
Cooperativeness

Results

Comparing tunable agents and fixed agents with empirical payoff matrices

		Pred. 2			
		C		D	
Pred. 1	C	0.792	0.792	0.592	0.296
	D	0.254	0.532	0.306	0.438

Tunable Agents

		Pred. 2			
		C		D	
Pred. 1	C	0.908	0.908	0.540	0.344
	D	0.340	0.600	0.436	0.468

Fixed Agents

Future Work

The tunable agent framework would be beneficial to any RL problem where there is some degree of uncertainty over the desired type of agent behaviour.

Wolfpack environment: One could train the prey to evade the predators and analyse the social dynamics.

Training scheme: One could easily use a different base RL algorithm to the DQN algorithm. Investigate the possibility of adapting the training methods for non-linear scalarisation.

Thank you for your attention.

Code available at:
https://github.com/docallaghan/tunable-agents