Options as responses: Grounding behavioural hierarchies in multi-agent RL

Alexander Sasha Vezhnevets[*1], **Yuhuai Wu** [*1,2], **Rémi Leblond**[1], **Joel Z. Leibo**[1]
[1] DeepMind, [2] University of Toronto

Abstract

We propose a novel hierarchical agent architecture for multi-agent reinforcement learning with concealed information. The hierarchy is grounded in the concealed information about other players, which resolves "the chicken or the egg" nature of option discovery. We factorise the value function over a latent representation of the concealed information and then re-use this latent space to factorise the policy into options. Low-level policies (options) are trained to respond to particular states of other agents grouped by the latent representation, while the top level (meta-policy) learns to infer the latent representation from its own observation thereby to select the right option. This grounding facilitates credit assignment across the levels of hierarchy. We show that this helps generalisation—performance against a held-out set of pre-trained competitors, while training in self- or population-play—and resolution of social dilemmas in self-play.

1 Introduction

By what principles can we break up agent policies into sub-policies to allow complex, hierarchical behaviour? How do we resolve the seemingly circular "chicken or the egg" nature of sub-policies discovery? If we have good sub-policies for a complex environment, then it is usually straightforward to form a strong overall policy out of them; conversely if we have a strong overall policy, it is often easy to divide this up into useful sub-policies. However, doing one without the other remains challenging – discovering good options can be as hard as solving the environment. This paper explores how *other agents* in multi-agent reinforcement learning provide sufficient structure to ground option discovery and resolve this "chicken or the egg" dilemma.

Discovering meaningful behavioural primitives and learning to use them to improve learning is the focus of hierarchical reinforcement learning (HRL). The options framework [31, 26] provides a canonical formalisation of a two-level HRL algorithm. The bottom level—an option—is a sub-policy, which takes in environment observations and outputs actions. An agent switches between options by using its policy-over-options (the top level). The framework doesn't specify any particular *grounding* for options—i.e. via which principles they break up the overall policies, which signal is to be used for switching between them and how they are to be learnt. Yet grounding is the key aspect by which different HRL methods vary. For instance, in feudal learning [7, 34], options are grounded in the agent's state space. The least restrictive grounding of [3] is strictly temporal, where options simply correspond to any policy followed for some time. Others have grounded options in tasks [11] or complex action spaces [23].

Focused on single agent case, these approaches still suffer from "chicken or the egg" paradox, where discovering options is as hard as solving the environment. This might explain why, despite recent progress, the state-of-the-art agents on the popular ATARI benchmark are flat [15, 36] – i.e. non hierarchical – and why the most impressive displays of the power of deep RL [30, 25, 35] did not use

[*] Authors contributed equally.

Preprint. Under review.

a hierarchical model either. In this work, we propose to explore a different grounding for options: concealed information about the states of other agents in multi-agent environments.

Consider this example – a modified game of rock-scissors-paper, where each strategy (e.g. rock), rather than being executed in one shot as an atomic action, requires a long sequence of actions that, taken as a whole, *constitutes* the chosen strategy. In keeping with the terminology introduced in [18], we call such a policy an *implementation* of the strategy. Both strategic decisions and their implementations must be learned by reinforcement learning.

We propose a stateful extension of the classic matrix game rock-paper-scissors, called Running With Scissors. In this *spatialised* extension, two players move around on a 2D grid, only observing a small window around their current location (the environment is *partially observed*). They can pick up resources (i.e. rocks, papers and scissors), which are scattered semi-randomly, and at any point decide to confront their opponent (see full details in Section 4.1). Once the confrontation occurs, or the episode times out, then the payoffs accruing to each player are calculated according to the standard anti-symmetric matrix formulation of rock-paper-scissors (as in, e.g., [13]). In this game, to implement a strategy—say the 'rock' pure strategy—an agent must seek out tiles with rock resources, while avoiding collecting non-rock resources, and then confront the opponent.

All the rich game theoretic structure of the rock-paper-scissors game is maintained in Running With Scissors. In addition, agents also have to cope with the complexity of learning implementations of strategies. For example, the game's intransitive response dynamics (rock beats scissors, which beat paper, which in turn beats rock), may induce endlessly recurring cyclic reinforcement learning challenges [4, 18, 24]. Furthermore, because they have imperfect information, agents cannot easily figure out their opponent's strategy and select the appropriate best response. However, they have ways to gather information about their opponent's intent. In Running With Scissors, agents are able to move around the map and observe what resources are missing, implying they were already collected by the adversary. A smart way to play this game is to first attempt to scout around and determine to what resource the adversary has committed itself, and then implement the best response to that strategy. The crucial point in this setup is that these sub-policies are all naturally grounded – the top level deals with the choice of strategy as a response to the opponent's strategy, and the bottom level with its implementation. They are also precisely what one hopes would be discovered by any option-discovery hierarchical reinforcement learning algorithm.

We cast the multi-agent learning problem as hierarchical RL and ground the hierarchy of behaviour in the latent representation, z, capturing the concealed information about *other* agents in the environment. We propose a hierarchical agent OPRE (OPtions as REsponses): at the low level of the hierarchy, each option is learning a best response policy to a particular configuration of z, while the top level of the hierarchy estimates z and thereby picks an option. We learn the latent representation z by factorising the value function over the concealed information using a mixture model. For training, we assume that we have access to this concealed information *in hindsight* for the purpose of generating learning signal;[2] we *never* use concealed information to generate any behaviour. This is similar to hindsight policy gradient [28] and experience replay [2] and other counterfactual RL works [6, 9, 21]. This grounding helps the agent distinguish between two types of mistakes: i) wrongly guessing the concealed information, thereby picking an inadequate strategy (e.g. rock vs paper) or ii) picking the right strategy, but executing it poorly. This way, the agent is capable of correctly assigning credit from strategic decisions and their execution to the corresponding part of the model – the policy *over* options (choice of strategy) versus the policy *of an* option (execution). This grounding resolves the "chicken or the egg" dilemma, by providing a clear training signal for the policy over options—the activations of latent variables that factorise the informed value function. To the best of our knowledge, this hierarchical approach grounded in opponents' state is unique; related works either do not assume a hierarchy of policies [9, 21, 14, 27], function with a centralised manager as top-level [1], or do not ground their hierarchy in game-theoretic fashion [33].

For experimental evaluation, we look at two measurements. **First** we propose to exploit the benefits of multi-agent environments, which opens richer possibilities in terms of measuring *generalisation* in reinforcement learning (as recently advocated [22, 12]). One such approach is to evaluate agents against diverse, pre-trained opponents that have not been encountered during training. In turn, training is performed via self-play or population-play, where agents respectively encounter copies of

[2]This is a reasonable assumption, akin to a sportsman watching a replay of the game to analyse their play a posteriori. StarCraft 2 players routinely watch replays with opponent play revealed during championships.

themselves, or agents of the same kind (architecture and learning method) training in parallel to them (see Section 5 for details). This measure of generalisation brings RL closer to the classic supervised learning setup, with clearly separates training and test sets. OPRE generalises better by separately extracting strategic and tactical knowledge of the game into separate levels of hierarchy. It then generates new behaviours by recombining learnt options given an unfamiliar opponent. Experiments show that OPRE significantly outperforms baselines on Running with Scissors, when evaluated against held-out opponents that the agents did not see in training. **Second**, we demonstrate that OPRE successfully resolves the social dilemma game Allelopathy [17] in self-play. In the game a high return for all the participants is possible if each occupies their niche. As [16, 19] showed, regular reinforcement learning agents in self-play suffer from a drive to greedily maximise their return, which leads to the collapse of the common good and, consequently, low return. In self-play this problem is accentuated by the fact that all agents are copies of each other, so if one does discover a free niche it will tend to immediately get swamped. We show that OPRE, due to its hierarchical representation of the policy, is able to dynamically choose the right niche and avoid the collapse of the common good.

The rest of the paper is organised as follows. In Section 2, we define the multi-agent RL setup formally, and we introduce the OPRE model in Section 3. We describe the test domains in Section 4 and present the experimental results in Section 5. We conclude in Section 6.

2 Multi-agent RL with concealed information

We consider an N-player partially observable Markov game $\mathcal{M}$ defined on a finite set of states $\mathcal{S}$. The observation function $\mathcal{X} : \mathcal{S} \times \{1, ..., N\} \to \mathbb{R}^d$, specifies each player?s d-dimensional view on the state space. In each state, each player i is allowed to take an action from its own set $\mathcal{A}^i$. Following their joint action $(a^1, ..., a^N) \in \mathcal{A}^1 \times ... \times \mathcal{A}^N$, the state changes obeys the stochastic transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}^1 \times ... \times \mathcal{A}^N \to \Delta(\mathcal{S})$ ($\Delta(\mathcal{S})$ denotes the set of discrete probability distributions over $\mathcal{S}$) and every player receives an individual reward defined as $r^i : \mathcal{S} \times \mathcal{A}^1 \times ... \times \mathcal{A}^N \to \mathbb{R}$ for player i. Finally, let $\mathcal{X}^i = \{\mathcal{X}(s, i)\}_{s \in \mathcal{S}}$ be the observation space of player i.

Each agent learns independently through their own experience of the environment by optimising for their own individual return, without direct communication to other agents. Notice that this is different from a large part of the recent work on multi-agent RL [9, 27, 14], where agents are working towards a common goal by optimising a shared reward (with the notable exception of [21], which allows for cooperative-competitive environments), often coordinating via an explicit communication channel.

The rest of the paper considers an "egocentric" view of one of the agents in the environment, thereby we will denote its observations and actions simply as x and a. The information concealed from the agent is denoted as x'. Here, we define it as the observations of all other agents. So for the agent of interest, $x' = \bigcup_{x^j \neq x} x^j$.

3 OPRE model

This section introduces the OPtions as REsponses (OPRE) model – a hierarchical agent for multi-agent RL. Figure 1 illustrates the structure of the model which consists of two parts. The first part (Figure 1, red box on the left) builds the latent space by factorising the value function over the concealed information, while the second part (Figure 1, blue box on the right) re-uses that latent space to factorise the agent's policy into options. Both value and policy are mixture models, where a categorical latent variable selects the mixture component. These mixture models are kept in sync via a KL term, which pulls the mixture weights to be similar in both for each step.

Building the latent space via value function factorisation. We model the value function of the agent as a mixture model, where the mixture weights $q(z|x'_t)$ depend on the concealed information x'_t, while the mixture components $c(x_{\leq t}, z)$ depend on the agent's own history of observations $x_{\leq t}$:

$$V_\pi(x_t, x'_t) = \sum_z q(z|x'_t) c(x_{\leq t}, z). \tag{1}$$

Here the latent variable z is assumed to be categorical and can be marginalised away. This construction compresses the concealed information into the distribution $q(z|x'_t)$ over the latent space z, but only focusing on the part which is *relevant to learning the value*.

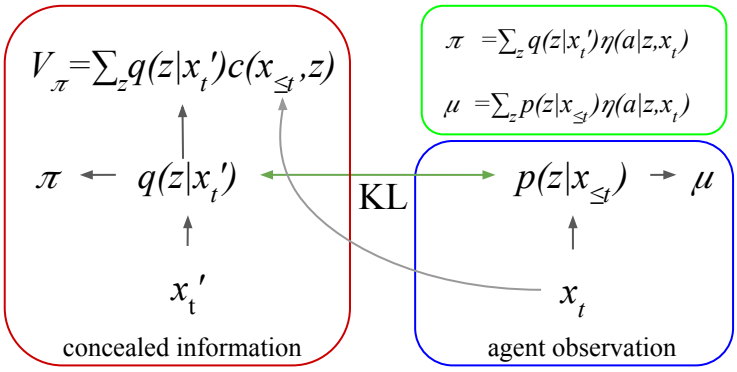

Figure 1: OPRE illustration. The first module (red box) uses concealed information, available in hindsight, to factorise the value function and target policy. The second module (blue box) produces behaviour policy, by approximating the latent distribution q.

Inducing options. The key idea is to use the same latent space to factorise the policy. Consider first this mixture policy:

$$\pi(a|x_t, x_t') = \sum_z q(z|x_t')\eta(a|x_t, z), \tag{2}$$

where $\eta(a|x_t, z)$ is a mixture component of the policy – an option. In this form, an option is a behavioural response to a particular latent representation z of concealed information x'. z corresponds to a group of opponent states for which a certain mixture component is activated (i.e. for which $q(z|x_t')$ is high). To compute $\pi(a|x_t, x_t')$ the agent needs to know $q(z|x_t')$, which depends on the concealed information x_t'. However, by definition, this quantity is not available to the agent. Therefore, we introduce the behaviour policy, which approximates it from the agent observation alone:

$$\mu(a|x_t) = \sum_z p(z|x_t)\eta(a|x_t, z). \tag{3}$$

In this way, $p(z|x_t)$ becomes the policy over options, which is learned separately as described below.

3.1 Learning

We use an IMPALA [8] based computational setup for training the agents. A few hundred CPU actors generate game trajectories, which are then batched and consumed by a GPU learner (one per unique agent). The learner receives temporally truncated sequences of 100 steps of trajectories in batches of 16. The parameters are periodically synced between the actors and the learner. The concealed information about other agents is only available to the learner (the actors, which generate behaviour, do not use concealed information). The pseudo-code is provided in the Appendix.

Learning the policy over options $p(z|x_t)$ is done via minimising $KL(q||p)$. The agent tries to learn the model of its opponents' behaviour through approximating $q(z|x_t')$ via $p(z|x_t)$. This makes learning significantly easier than in single agent HRL by avoiding the "chicken or the egg" issue, since there is a clearly defined target q for the policy over options p.

Learning options and the value function. During training, on the learner we can compute both μ and π by simply marginalising over z via summation, since z is categorical. We propose to use the policy gradient of π for training options, since it corresponds to using a better policy over options q. This avoids the "chicken or the egg" issue again by using the fully informed q as policy over options. Since q captures all that is necessary from the concealed information x' to factorise the value, it should be a good basis for factoring the policy into options as well. In other words, we are training options for a "perfect" policy over options q and then learning to approximate it with p. We use V-trace [8] for off-policy correction of the policy gradient updates and for learning the value function. That is we generate trajectories by following the behaviour policy, μ, but we learn (off-policy) the mixture-components of policy, π, using policy gradients. Importantly, we do not

propagate gradients from the policy gradient into q, completely leaving its training up to the KL and value function gradients. We also emphasise that we only use the concealed information x' for learning and never use it to run the agent's policy.

Regularisation. We want to encourage options to be smooth in time within an episode (switch less) and diverse across episodes. To this end, we use the following regularisation loss: $H^T(q) - H^B(q) - H(\pi)$. The first two terms correspond to the maximisation of the marginal entropy of q over a batch and the minimisation of its marginal entropy over time. The last term maximises the entropy of the target policy to encourage exploration and avoid policy collapse.

3.2 Credit assignment in multi-agent RL

Notice, how multi-agent aspect complicates credit assignment as the outcomes the agent is learning from are the product of complex interactions between its own states and actions and the states and actions of others. The same actions by an agent could lead to both high and low reward, depending on the opponents' actions. For example, if agent chooses to play a 'rock' strategy against an opponent who plays 'paper', rather than correcting its 'rock' option to be more like 'scissors', we only want it to change its decision to play 'rock' in the first place. Flat agents are not capable of separating these two signals and are likely to cycle between different strategies, as discussed in [5]. Notice how OPRE resolves this by separately learning to credit the policy over options p for selecting strategies and the options η for executing them. Indeed, if p is far from q then the off-policy term from V-trace will zero the gradient for η and the only training signal will be the KL between p and q.

4 Domains

All test domains are partially observable, spatially and temporally extended multi-agent games with a game-theoretic payoff structure. Contrary to a large amount of multi-agent studies, the tasks are not cooperative (agents do not share a reward signal) and do not feature a communication channel. The Appendix contains some visualisations of the domains, that will help understanding.

4.1 Running with scissors (RWS)

RWS is a spatial version of the rock-scissors-paper game, where agents get to collect the items in a grid world with partial observability and then play them out against each other. Formally, it is a Markov game [20] – Markov games are to matrix games as Markov decision processes are to bandit problems. The grid has dimensions 13×21; each of the two players starts at a random position on the grid and can observe a 4×4 area around itself. There are 36 evenly spaced resources (i.e. rock, scissors or paper) tiles on the grid, 18 of which always have the same resource while others get a random one. When a player steps on a tile with the resource, it picks it up – the resource is removed from the grid and is allocated into the player's inventory v, which is initialised to contain one of each resource type. Additionally, players receive specialisation bonus, that is that when a resource is picked up, the amount of that resource in their inventory becomes $v_t[i] = 1.1v_{t-1}[i] + 1$, where $v_{t-1}[i]$ is the amount of resource i they had before. This is an incentive for specialisation and punishes agents for picking a little bit of everything.

There are two termination conditions for the episode: i) timeout of 1000 steps or ii) one agent tags another. Tagging is a special action that highlights a 3×3 area in front of the tagging player and is successful if the other player is in that area. Which agent was tagged does not matter; the outcome is the termination of the episode and scoring. The rewards accruing to each player are calculated according to the standard anti-symmetric matrix formulation of rock-paper-scissors (as in, e.g., [13]) — $r^0 = v^0 M (v^1)^T$, where v^0 and v^1 are the respective inventories and M is the anti-symmetric matrix of the game. To get a high reward an agent should correctly identify what it is that its opponent is collecting (e.g. rock) and collect a counter (e.g. paper). In addition, the rules of the game are not assumed given; the agents must explore to discover them. Thus it is simultaneously a game of *imperfect* information—each player possesses some private information not known to their adversary (as in e.g. poker [29])—and *incomplete* information—lacking common knowledge of the rules [10].

4.2 Allelopathy

The Allelopathy game is described in detail in [17] and has two main rules. (1) shrubs grow in random positions on an open field. Shrubs "allelopathically" suppress one another's growth in a local region of radius 2. That is, the probability that a seed of a given type grows into a shrub in any given timestep is inversely proportional to the number of nearby shrubs of other types. (2) Agents can eat either type of shrub. However, switching frequently between digesting different shrub types imposes a switching cost, since this makes them harder to digest. Thus, agents benefit from specialising in eating only a single type of shrub. Agents receive increasing rewards for repeatedly harvesting the same type of shrub (up to a maximum of $r = 256$). Rewards drop back down to their lowest level, $(r = 1)$, when the agent harvests a different type of shrub (paying the switching cost). Importantly, one of the shrubs is less valuable than another, with maximum reward of only 8. Thus an agent that randomly harvests any shrub they come across is likely to receive low rewards. An agent that only harvests a particular type of shrub while ignoring others will obtain significantly greater rewards. The combined effect of these rules is to make it so that a specialist in any one shrub type benefits from the presence of others who specialise in different shrub types since their foraging increases the growth rate of all the shrubs they do not consume. Yet specialists in more valuable shrub will reap immensely higher rewards. Thereby we create a social dilemma – there exists an equilibrium state with high rewards for everyone (albeit less for some), but egoistically each agent would benefit from harvesting the expensive shrub, which creates danger of over-harvesting and ecosystem collapse.

5 Experiments

This section presents our experimental results on the domains described above. We aim to validate that OPRE has superior generalisation through the grounding of its behavioural hierarchy. The next paragraph explains the baselines we compare to. We then describe the training and evaluation regimes, and present the actual results. Please refer to the Appendix for more details on networks design and the exact values of hyper-parameters.

Baselines. We use IMPALA [8] to train all of our agents (see section 3.1). V-trace is used to correct for off-policy due to lag of parameter synchronisation on actors and learners. Our first baseline uses a standard architecture with a convolutional network base and a LSTM on top, with linear policy and value head. The second baseline is designed to test whether improvement can be achieved by conditioning the value function (but not the policy) on the concealed information. We call it the **baseline oracle V**. This is very similar to works [21, 9], which use a critic with access to concealed information at training time and do not have a hierarchical structure – although their focus is mostly in the collaborative setup, where agents share the reward. Concealed information is processed in the same way as in OPRE and then concatenated to the value head's input. The last baseline is a version of OPRE, in which additional policy gradient from the behavioural policy μ is used for training options; coined **OPRE mix PG**. The hyper-parameters were tuned on RWS in the regime of training and evaluating against competitors; we first tuned the parameters of the baseline, then tuned the extra hyper-parameters of OPRE (see the Appendix for all hyper-parameters and their values).

Training regimes. We use three training regimes. First, *self-play*, where during training the agent is playing with copies of itself (however many the game requires). Second, *population-play*, where the agent is playing with a population of 6 agents with shared architecture, but each member of the population has its own parameters which it trains individually. For some experiments with RWS we also train directly against a separately pre-trained set of competitors. This setup is very similar to multi-task learning. In all cases we run for $5 \cdot 10^8$ learning steps.

Running with scissors is where we focus on measuring generalisation to a held-out set of pre-trained competitors, which were generated as follows. We trained a set of 14 agents in simultaneous population-play.[3] 12 of them were augmented with pseudorewards during training that incentivised them to learn a particular pure strategy. For example, four agents implementing 'rock' pure strategies were obtained using a pseudoreward of $+10$ for each rock resource collected and -5 for each non-rock resource collected. Eight more agents, four of them implementing 'paper' and four implementing 'scissors' pure strategies were obtained analogously. The remaining two out of the 14 simultaneously

[3] As the self-play setup was too hard for any of the methods to take off, we omit the results to save space.

learning agents were given no pseudorewards for robustness. We use separate CPU jobs that evaluate current networks against the pre-trained agents in parallel throught the training.

We compare the performance of OPRE and the baselines in two settings: i) trained directly against the held-out set – measuring reward fitting power; ii) trained in population-play, evaluated on the held-out set – measuring generalisation. This lays bare if the methods overfit, i.e. perform well in the first case, but not in the second. Figure. 2 plots the respective reward curves.

First, we notice that OPRE generalises from population-play to held-out competitors best, with *5 times higher* average return than the baseline. Second, while OPRE with mixed policy gradient is the best method for training directly against the hold-out, it drops to second place when generalisation is required. One hypothesis is that using a 'perfect' policy over options yields more delineated options, whereas options learnt over p are less specialised (as they receive learning signal even when p picked the wrong z). Third, we see that the baseline whose value function is conditioned on concealed information flips from being second best (if trained directly against the held-out) to being the worst if required to generalise, getting zero average return. This demonstrates that conditioning the value function on concealed information (as in [9, 14, 27]), might induce overfitting. Qualitatively, oracle V baseline learns an unsystematic policy, which collects random resources and engages the opponent at the first opportunity, without any responsiveness to the opponent. In contrast OPRE, is sensitive to information that reveals opponents strategy – e.g. if it notices that rocks are gone, it picks paper. OPRE is less directly exposed to privileged information, which results in better generalisation properties. Further, although the 'oracle' value function can account for the opponent's state, the flat policy cannot correctly represent the multi-modality of the required response. In comparison, OPRE explicitly models the latent state of the opponent and can reuse this latent space to efficiently factor its policy.

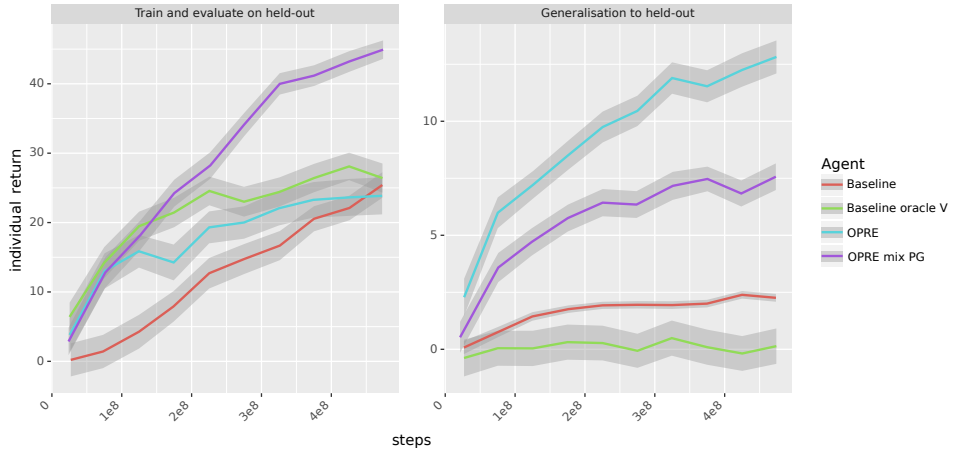

Figure 2: Running with scissors. Reporting average individual return at a training step with confidence intervals of 0.95, averaged over the population. Left: training and evaluating on held-out set; Right: training in population-play and evaluating on the held-out set. Notice overfitting in the baseline with access to concealed information and OPRE with mixed policy gradient.

Allelopathy allows us to evaluate whether OPRE agents are better at finding an ecological niche in a non-zero-sum game, where the optimality of an agent's individual policy depends on the behaviour of others. In allelopathy, it pays to specialise in whatever is in low demand. Miss-balance between the value of shrubs creates a drive to over-harvest the more valuable one. A good policy tracks the demand on shrub types (which depends on others) and strikes the trade-off between value of the shrub and its presence just right. In population-play, this specialisation can be established slowly, over the episodes across different agents, such that each agent picks a niche and stays in it. Self-play is, on the other hand, much more challenging since the policy has to be able to specialise within one episode by observing the opponents and adjusting its niche accordingly. As for RWS, we have pre-trained 6 competitors that received a negative pseudo-reward for eating a less rewarding shrub, thereby making them specialist in the most rewarding shrub. Evaluating against them probes whether an agent can identify that a niche of expensive shrub is taken and specialise correctly on the cheaper

one. Figure 3 shows results for self-play and Figure 4 for population-play. In population-play, OPRE is second best in terms of collective return in training and is the best when generalising to the hold-out set. The oracle V baseline demonstrates a similar overfitting behaviour as on RWS, getting the best collective return in training, but marginally under-performing on held-out competitors. In self-play OPRE demonstrates a stable, high return in both training and generalisation. Notice how the oracle V baseline is unstable – oscillating between 6 and 12 thousand points, unlike OPRE, which monotonically increases its return. OPRE mix PG underperforms, validating the choice of policy gradient for training options. Interestingly, the gains demonstrated by OPRE over the baselines are bigger in the harder case of self-play, where a single policy needs to be able to represent several meaningful policies and choose the right one in a single episode, than the easier one of population-play where specialisation can happen over the course of many episodes, and it is the model architecture (not a single policy) that needs to be able to represent several strategies.

Figure 3: Allelopathy, trained in self-play. Left: average collective return at a training step; Right: average individual return at training step, evaluated against held-out competitors.

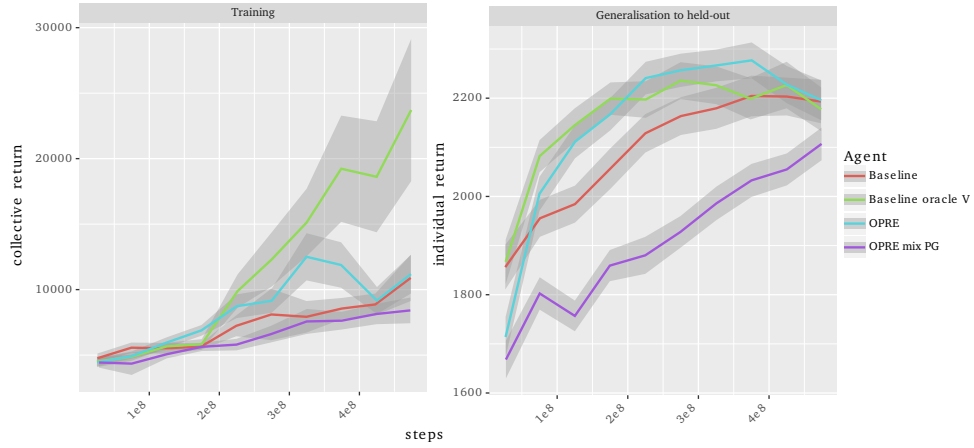

Figure 4: Allelopathy, trained population-play. Left: average collective return at a training step; Right: average individual return at training step, evaluated against held-out competitors.

6 Conclusions

The options framework suffers from the fact that discovering meaningful options is often as hard as solving the initial environment. We have proposed a solution to this issue by grounding the option selection in the concealed information of other agents in multi-agent environments. We introduced

OPRE – a new agent, which factorises the value function over the latent representation of other agents' concealed information, and then reuses this latent representation to factor the policy into options. It only requires access to concealed information for learning the value function during training and never uses it to generate behaviour. We demonstrated improved performance above strong baselines on two domains: the newly introduced Running with Scissors and a social dilemma Allelopathy.

Our key insight is that a policy can be factored into options by re-using a factorisation of the value function over concealed information, if it is available during training. As many domains fit this description, this insight may have implications beyond multi-agent environments. One exciting direction for future work is extending OPRE to multi-task learning with concealed task definition, where the agent has to both figure out which task it is solving as well as how to solve it. Another promising idea consists in performing the factorisation not only on the value function, but also on the environment dynamics through, for example, generalised value functions [32].

References

[1] Sanjeevan Ahilan and Peter Dayan. Feudal multi-agent hierarchies for cooperative reinforcement learning. *arXiv preprint arXiv:1901.08492*, 2019.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[3] Pierre-Luc Bacon, Doina Precup, and Jean Harb. The option-critic architecture. In *AAAI*, 2017.

[4] David Balduzzi, Karl Tuyls, Julien Perolat, and Thore Graepel. Re-evaluating evaluation. In *Advances in Neural Information Processing Systems*, pages 3268–3279, 2018.

[5] David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech M Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. *arXiv preprint arXiv:1901.08106*, 2019.

[6] Lars Buesing, Theophane Weber, Yori Zwols, Nicolas Heess, Sebastien Racaniere, Arthur Guez, and Jean-Baptiste Lespiau. Woulda, coulda, shoulda: Counterfactually-guided policy search. In *International Conference on Learning Representations*, 2019.

[7] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *NIPS*. Morgan Kaufmann Publishers, 1993.

[8] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. *ICML*, 2018.

[9] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[10] John C Harsanyi. Games with incomplete information played by "bayesian" players, i–iii part i. the basic model. *Management science*, 14(3):159–182, 1967.

[11] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[12] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] Josef Hofbauer and Karl Sigmund. Evolutionary game dynamics. *Bulletin of the American Mathematical Society*, 40(4):479–519, 2003.

[14] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. *arXiv preprint arXiv:1810.02912*, 2018.

[15] Steven Kapturowski, Georg Ostrovski, Will Dabney, John Quan, and Remi Munos. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=r1lyTjAqYX`.

[16] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 464–473. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[17] Joel Z Leibo, Julien Perolat, Edward Hughes, Steven Wheelwright, Adam H Marblestone, Edgar Duéñez-Guzmán, Peter Sunehag, Iain Dunning, and Thore Graepel. Malthusian reinforcement learning. *arXiv preprint arXiv:1812.07019*, 2018.

[18] Joel Z Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv preprint arXiv:1903.00742*, 2019.

[19] Adam Lerer and Alexander Peysakhovich. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *arXiv preprint arXiv:1707.01068*, 2017.

[20] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[21] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, 2017.

[22] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[23] Josh Merel, Arun Ahuja, Vu Pham, Saran Tunyasuvunakool, Siqi Liu, Dhruva Tirumala, Nicolas Heess, and Greg Wayne. Hierarchical visuomotor control of humanoids. In *International Conference on Learning Representations*, 2019.

[24] Shayegan Omidshafiei, Christos Papadimitriou, Georgios Piliouras, Karl Tuyls, Mark Rowland, Jean-Baptiste Lespiau, Wojciech M Czarnecki, Marc Lanctot, Julien Perolat, and Remi Munos. $\{\backslash alpha\}$-rank: Multi-agent evaluation by evolution. *arXiv preprint arXiv:1903.01373*, 2019.

[25] OpenAI. Openai five. `https://blog.openai.com/openai-five/`, 2018.

[26] Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts, 2000.

[27] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, 2018.

[28] Paulo Rauber, Avinash Ummadisingu, Filipe Mutz, and Juergen Schmidhuber. Hindsight policy gradients. *arXiv preprint arXiv:1711.06006*, 2017.

[29] Tuomas Sandholm. Solving imperfect-information games. *Science*, 347(6218):122–123, 2015.

[30] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[31] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 1999.

[32] Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *AAMAS*, 2011.

[33] Hongyao Tang, Jianye Hao, Tangjie Lv, Yingfeng Chen, Zongzhang Zhang, Hangtian Jia, Chunxu Ren, Yan Zheng, Changjie Fan, and Li Wang. Hierarchical deep multiagent reinforcement learning. *arXiv preprint arXiv:1809.09332*, 2018.

[34] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3540–3549. JMLR. org, 2017.

[35] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M. Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Yuhuai Wu, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019.

[36] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288, 2017.

A Network designs

Given the perceptual simplicity of environments, we use a very slim architecture. We first describe the baseline and then build from it. The network starts with a 1D convolutional layer with 6 channels, followed by an MLP with $(64, 64)$ neurons, then by an LSTM with 128 hidden neurons, and concluding with two linear heads for policy and value. For OPRE we have 16 policy heads, where each is an MLP with 128 hidden neurons. We use the same convolutional network with a new MLP on top to produce q from the concealed information. The policy over options p is produced via a linear layer from LSTM activations. For Allelopathy, which has several other agents in the game, we process each of their observations with the same convolutional net and then pool using summation.

B Hyper-parameters

We use discount of 0.99 for all games. The learning rate was set to 0.0004, and the weight of entropy regularisation of the policy logits for all method was 0.003. The weight of entropy regularisation for q was 0.001. The hyper-parameters were tuned on RWS in the regime of training and evaluating against competitors; we first tuned the parameters of the baseline, then tuned the extra hyper-parameters of the OPRE.

C Optimization

We used RMS-prop optimiser with the following parameters: learning rate=0.0004, ϵ=1e-5, momentum=0.0, decay=0.99

D Domain illustrations

Here we provide illustrations of the domains. Figure 5 visualises the initial grid configuration of Running with Scissors and Figure 6 of Allelopathy.

E Pseudo code

Algorithm 1 presents the pseudo code for learners and actors for training OPRE agents. In population play, we launched 6 learners and 1500 actors, in parallel. In self-play only one learner and 200 actors.

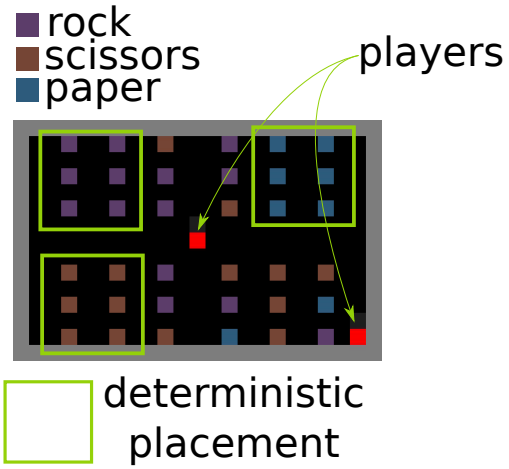

Figure 5: Running with scissors. This is the initial configuration of the grid. Players spawn at any random location, except for those occupied by resources. Some resources are placed deterministicaly (shown by green rectangles), others spawn randomly.

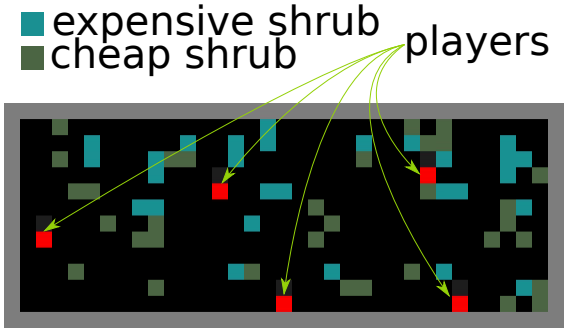

Figure 6: Allelopathy. This is the initial configuration of the grid. Players spawn at any random location, except for those occupied by shrubs. Shrub locations are initialised randomly.

Algorithm 1 Training OPRE

procedure ACTOR(N, K) // takes max number of steps and number of agents
  while $n < N$ **do**
    if *time to sync* **or** $n = 0$ **then**
      $\Theta = \{\theta_k\}_{k=1}^{K} \leftarrow \Theta_{step}$ // every once in a while, sync parameters with learner
    $i \leftarrow$ *random*(K), $j \leftarrow$ *random*(K) // draw two agents to play at random
    $\tau \leftarrow$ *Play*$(\mu_{\theta_i}, \mu_{\theta_j})$ // generate trajectory by playing behaviour policies
    Queue(τ) // queue the trajectory for the learner
    $n \leftarrow n_{learner}$ // sync learning step counter with the learner
procedure LEARNER(N, k, T, B) // takes max number of steps, agent index, batch dimensions
  $\theta \leftarrow$ *random*$()$
  $n \leftarrow 0$
  while $n < N$ **do**
    $\beta \leftarrow$ *Dequeue*(k, B, T) // get a batch for a respective agent of $B \times T$ size
    $\nabla\pi_\theta, \nabla V_\theta \leftarrow$ *V-trace*(β, θ) // run target policy π and get gradients with V-trace [8]
    $\nabla Reg \leftarrow \nabla(H^T(q_\theta) - H^B(q_\theta) - H(\pi_\theta))$ // gradients from regularisation
    $\theta \leftarrow$ *BackProp*$(\nabla\pi_\theta + \nabla V_\theta + \nabla KL(q_\theta || p_\theta) + \nabla Reg)$ // update parameters with backprop
    $n \leftarrow n + T \cdot B$ // increment step