Multi-Agent Common Knowledge Reinforcement Learning

Jakob N. Foerster[*†] **Christian A. Schroeder de Witt**[*†]

Gregory Farquhar[†] **Philip H. S. Torr**[†] **Wendelin Boehmer**[†] **Shimon Whiteson**[†]

Abstract

In multi-agent reinforcement learning, centralised policies can only be executed if agents have access to either the global state or an instantaneous communication channel. An alternative approach that circumvents this limitation is to use centralised training of a set of decentralised policies. However, such policies severely limit the agents' ability to coordinate. We propose *multi-agent common knowledge reinforcement learning* (MACKRL), which strikes a middle ground between these two extremes. Our approach is based on the insight that, even in partially observable settings, subsets of agents often have some *common knowledge* that they can exploit to coordinate their behaviour. Common knowledge can arise, e.g., if all agents can reliably observe things in their own field of view and know the field of view of other agents. Using this additional information, it is possible to find a centralised policy that conditions only on agents' common knowledge and that can be executed in a decentralised fashion. A resulting challenge is then to determine at what level agents should coordinate. While the common knowledge shared among all agents may not contain much valuable information, there may be subgroups of agents that share common knowledge useful for coordination. MACKRL addresses this challenge using a hierarchical approach: at each level, a controller can either select a joint action for the agents in a given subgroup, or propose a partition of the agents into smaller subgroups whose actions are then selected by controllers at the next level. While action selection involves sampling hierarchically, learning updates are based on the probability of the joint action, calculated by marginalising across the possible decisions of the hierarchy. We show promising results on both a proof-of-concept matrix game and a multi-agent version of StarCraft II Micromanagement.

Introduction

Cooperative multi-agent systems are ubiquitous, for example, in the coordination of autonomous cars (Cao et al., 2013). However, how to learn control policies for such systems remains a major open question. One approach is to learn centralised policies that select joint actions conditioned on the global state or joint observation. However, in order to execute such policies, the agents must have access to either the global state or an instantaneous communication channel with sufficient bandwidth to enable them to aggregate their individual observations. These requirements often do not hold in practice but even when they do, learning a centralised policy is often infeasible as the size of the joint action space grows exponentially in the number of agents.

These difficulties motivate an alternative approach: centralised training of decentralised policies. During learning the agents can share observations, parameters, gradients, etc. without restriction but the result of learning is a set of decentralised policies such that each agent can select actions based only on its individual observations. While significant progress has been made in this direction (Foerster et al., 2016, 2017, 2018; Jorge, Kageback, and Gustavsson, 2016; Kraemer and Banerjee, 2016; Rashid et al., 2018), the requirement that policies must be fully decentralised severely limits the agents' ability to coordinate their behaviour. Often agents are forced to ignore information in their individual observations that would in principle be useful for maximising reward, because acting on it would make their behaviour less predictable to their teammates.

In this paper, we propose *multi-agent common knowledge reinforcement learning* (MACKRL), which strikes a middle ground between these two extremes. The key insight is that, even in partially observable settings, subsets of the agents often possess some *common knowledge* that they can exploit to coordinate their behaviour. Common knowledge for a set of agents consists of facts that all agents know and "each individual knows that all other individuals know it, each individual knows that all other individuals know that all the individuals know it, and so on" (Osborne and Rubinstein, 1994).

We formalise a multi-agent setting that suffices to give rise to common knowledge. The setting involves assumptions that, while restrictive, are naturally satisfied in a range of multi-agent problems. Intuitively, common knowledge can arise between two agents when each agent can observe the other, and doing so disambiguates what the other has observed. For example, if each agent can reliably observe things that are within its field of view and the agents know each other's fields of view, then two agents possess common knowledge whenever they see each other. Such a scenario, illustrated in Figure 1, arises in tasks such as robo-soccer (Genter, Laue, and Stone, 2017) and multi-agent StarCraft Micromanage-

[*]Equal contribution. Correspondence to Christian Schroeder de Witt <cs@robots.ox.ac.uk>

[†]University of Oxford, UK

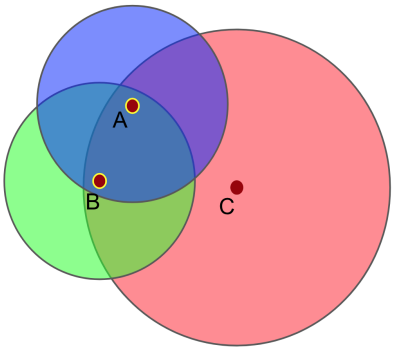

Figure 1: Three agents and their fields of view. A and B's locations are common knowledge to A and B as they are within each other's fields of view. However, even though C can see A and B, it shares no common knowledge with them.

ment (Synnaeve et al., 2016). It could also arise in a range of real-world scenarios, e.g. a self-driving car that knows which sensors other autonomous vehicles that it observes are equipped with.

Common knowledge is useful because, by definition, each agent in a group can independently deduce the common knowledge shared by that group. Consequently, a centralised joint policy that conditions only on the common knowledge of all agents can be executed in a decentralised fashion: each agent simply queries the centralised policy for a joint action and then executes their part of that joint action. Since each agent supplies the same common knowledge as input, they return the same joint action, yielding coordinated behaviour.

However, the presence of common knowledge gives rise to a major challenge. Smaller groups of agents often possess more common knowledge than larger groups, making it unclear at what level agents should coordinate. On the one hand, coordination would ideally happen globally, that is, a fully centralised policy would select a joint action for all agents. However, the common knowledge shared among all agents may be weak, limiting what such a policy can condition on. On the other hand, the agents could be broken into subgroups that share more common knowledge within them. Coordination would no longer occur across the subgroups, but action selection within each group could condition on a richer common knowledge signal.

MACKRL addresses this challenge using a hierarchical approach. At each level of the hierarchy, a controller can either select a joint action for the agents in its subgroup, or propose a partition of the agents into smaller subgroups, whose actions are then selected by controllers at the next level of the hierarchy. During action selection, MACKRL simply samples sequentially from the hierarchy of common knowledge controllers. However, during training, the total probability of the joint action is evaluated by marginalising over all choices that could have been taken at every level of the hierarchy. Thus, even the parameters governing a subgroup that was not selected during the action sampling can receive gradients.

Using a matrix game, we show results for a tabular version of MACKRL that outperforms both independent and joint-action learners, the code is in the Supplementary Material. Furthermore, we develop a Deep RL variant called *pairwise MACKRL*, which uses pairwise coordinated strategies implemented with RNN policies and a centralised critic (Foerster et al., 2018). On a challenging multi-agent variant of StarCraft II Micromanagement pairwise MACKRL outperforms a baseline agent which uses a centralised critic.

Related Work

Multi-agent reinforcement learning (MARL) has been studied extensively in small environments (Busoniu, Babuska, and De Schutter, 2008; Yang and Gu, 2004), but scaling it to large state spaces or many agents has proved problematic. One reason is that the joint action space grows exponentially in the number of agents, making learning intractable. Guestrin, Koller, and Parr (2002) propose the use of *coordination graphs*, which exploit conditional independence properties between agents that are captured in an undirected graphical model, in order to efficiently select joint actions. *Sparse cooperative Q-learning* (Kok and Vlassis, 2004) also uses coordination graphs to efficiently maximise over joint actions in the Q-learning update rule. Whilst these approaches allow agents to coordinate optimally, they require the coordination graph to be known and for the agents to either observe the global state or to be able to freely communicate. In addition, in the worst case there are no conditional independencies to exploit and maximisation must still be performed over an intractably large joint action space.

There has been much work on scaling MARL to handle complex, high dimensional state and action spaces. In the setting of fully centralised training and execution, Usunier et al. (2016) frame the problem as a *greedy MDP* and train a centralised controller to select actions for each agent in a sequential fashion. Sukhbaatar, Szlam, and Fergus (2016) utilise a network architecture that enables the agents to share information before selecting their actions centrally. Peng et al. (2017) train a fully centralised controller that utilises a bidirectional RNN as a communication channel between agents, and decompose the global reward into per-agent local rewards. These approaches assume there is unlimited bandwidth available to communicate between agents.

One way to decentralise the agents' policies is to learn a separate Q-function for each agent as in *independent Q-learning* (Tan, 1993). Foerster et al. (2017) and Omidshafiei et al. (2017) examine the problem of instability that arises from the perceived non-stationarity of the environment induced by both the agents' exploration and their changing policies. Rashid et al. (2018) and Sunehag et al. (2017) propose learning a centralised value function that factors into per-agent components to allow for decentralisation. Gupta, Egorov, and Kochenderfer (2017) learn separate policies for each agent in an actor-critic framework, where the critic for each agent conditions only on per-agent information. Foerster et al. (2018) and Lowe et al. (2017) propose a single centralised critic that conditions on all available information during training and is used to train decentralised actors. None of these approaches explicitly learn a policy over joint actions and hence are limited in the coordination they can achieve.

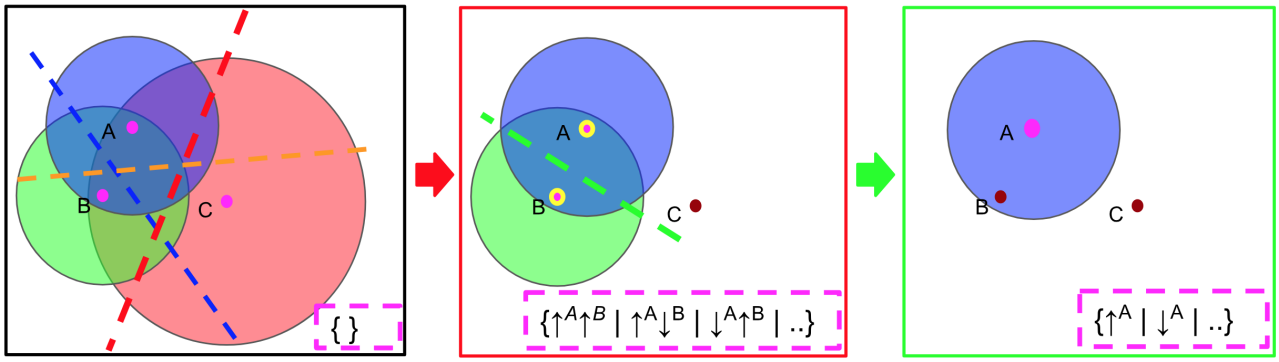

Figure 2: Pairwise MACKRL. Left: the pair selector must assign agents to pairs (plus a singleton in this case). Middle: the pair controller can either partition the pair or select among the pair's joint actions; Right, at the last level, the controller must select an action for a single agent.

Genter, Agmon, and Stone (2013) and Barrett, Stone, and Kraus (2011) examine *ad hoc teamwork*: how agents can cooperate with previously unseen teammates when there are a variable number of non-learning agents. Albrecht and Stone (2017) address ad hoc teamwork by maintaining a belief over a set of parameterised hypothetical behaviours and updating them after each observation. Panella and Gmytrasiewicz (2017) treat the behaviours of teammates as a stochastic process and maintain beliefs over these in order to learn how to coordinate with previously unseen agents. Makino and Aihara (2006) develop an algorithm that reasons over the beliefs over policies of other agents in fully observable settings. In ad-hoc teamwork, the learning agents learn to coordinate with other agents with fixed, non-learning behaviour; in our setting all agents learn at the same time, with the same known algorithm.

Thomas et al. (2014) explore the psychology of common knowledge and coordination. Rubinstein (1989) shows that any finite number of reasoning steps, short of the infinite number required for common knowledge, can be insufficient for achieving coordination. Korkmaz et al. (2014) examine common knowledge in scenarios where agents use Facebook-like communication and show that a complete bipartite graph is required for common knowledge to be shared amongst a group. Brafman and Tennenholtz (2003) use a common-knowledge protocol to improve coordination in common interest stochastic games but, in contrast to our approach, establish common knowledge about agents' action sets and not about subsets of their observation spaces.

Aumann and others (1974) introduce the concept of a *correlated equilibrium*, whereby a shared *correlation device* helps agents coordinate better. Cigler and Faltings (2013) examine how the agents can reach such an equilibrium when given access to a simple shared *correlation vector* and a communication channel. Boutilier (1999) augments the state space with a coordination mechanism, to ensure coordination between agents is possible in a fully observable multi-agent setting. This is in general not possible in the partially observable setting we consider. Instead of relying on a shared communication channel or full observability, MACKRL achieves coordination by utilising common knowledge.

Amato, Konidaris, and Kaelbling (2014) propose MacDec-POMDPs, which use hierarchically optimal policies that allow agents to undertake temporally extended macro actions. Liu et al. (2017) investigate how to learn such models in environments where the transition dynamics are not known. Makar, Mahadevan, and Ghavamzadeh (2001) extend the MAXQ single-agent hierarchical framework by Dietterich (2000) to the multi-agent domain. They allow certain policies in the hierarchy to be *cooperative*, which entails learning the joint action-value function and allows for faster coordination across agents. However, unlike MACKRL this requires the agents to communicate during execution.

Kumar et al. (2017) use a hierarchical controller that produces subtasks for each agent and chooses which pairs of agents should communicate in order to select their actions. In contrast with our approach, they allow communication during execution, and do not test on a sequential decision making task.

Problem Setting

Cooperative multi-agent tasks can be modelled as *decentralised partially observable Markov decision processes* (Dec-POMDPs) (Oliehoek, Spaan, and Vlassis, 2008).

In this paper, we consider a particular kind of Dec-POMDP that gives rise to common knowledge among agents. In this Dec-POMDP, the state of the system $s \in \mathcal{S}$ is composed of a number of entities $e \in \mathcal{E}$, with state features s^e, i.e., $s = \{s^e \,|\, e \in \mathcal{E}\}$.

A subset of the entities are agents: $a \in \mathcal{A} \subseteq \mathcal{E}$, where $|\mathcal{A}| = n$. Other entities could be enemies, obstacles, or goals.

At each timestep, each agent can select an action $u^a_{\text{env}} \in \mathcal{U}^a_{\text{env}}(s)$, where the subscript indicates an environment action. Given a joint action $\mathbf{u}_{\text{env}} := (u^1_{\text{env}}, \ldots, u^n_{\text{env}}) \in \mathcal{U}_{\text{env}}$, the discrete-time system dynamics draw the successive state $s' \in \mathcal{S}$ from the conditional distribution $P(s'|s, \mathbf{u}_{\text{env}})$ and yield a joint reward from the function $r(s, \mathbf{u}_{\text{env}})$.

The agents have a particular form of partial observability. At each time step, each agent a receives an observation $z^a \in \mathcal{Z}$ containing the subset of state features s^e from all the entities e that a can see. Whether a can observe e is determined by the binary mask $\mu^a\left(s^a, s^e\right) \in \{\top, \bot\}$ over the agent's and entity's observable features. An agent can always observe itself, i.e., $\mu^a(s^a, s^a) = \top, \forall a \in \mathcal{A}$. The set of all entities the agent can see is therefore $\mathcal{M}^a_s :=$

$\{e \,|\, \mu^a(s^a, s^e)\} \subseteq \mathcal{E}$, and the agent's observation is specified by the deterministic observation function $o(s, a)$ such that $z^a = o(s, a) = \{s^e \,|\, e \in \mathcal{M}_s^a\} \in \mathcal{Z}$.

The goal of the agents is to maximise the discounted return $R_t = \sum_{l=0}^{H} \gamma^l \, r(s_{t+l}, \mathbf{u}_{t+l,\text{env}})$ experienced during trajectories of length H. The joint policy $\pi(\mathbf{u}_{\text{env}}|s)$ is restricted to a set of decentralised policies $\pi^a(u^a_{\text{env}}|\tau^a)$, that can be executed independently, i.e. each agent's policy conditions only on its own action-observation history τ^a. We denote a policy across the joint action space $\mathcal{U}^{\mathcal{G}}_{\text{env}}$ of the group $\mathcal{G} \subseteq \mathcal{A}$ of agents as $\pi^{\mathcal{G}}$. Following previous work (Foerster et al., 2016, 2017, 2018; Jorge, Kageback, and Gustavsson, 2016; Kraemer and Banerjee, 2016; Rashid et al., 2018), we allow centralised learning of decentralised policies.

Our setting is thus a special case of the Dec-POMDP in which the state space factors into entities and the observation function is deterministic, yielding perceptual aliasing in which the state features of each entity are either accurately observed or completely occluded. In the next section, we leverage these properties to establish common knowledge among the agents. The Dec-POMDP could be augmented with additional state features that do not correspond to entities, as well as additional possibly noisy observation features, without disrupting the common knowledge we establish about entities. For simplicity, we omit such additions.

Common Knowledge

A key property of the binary mask μ^a is that it depends only on the features s^a and s^e to determine whether agent a can see entity e. If we assume that all agent a's mask μ^a is common knowledge, then this means that another agent b, that can see a and e, i.e., $a, e \in \mathcal{M}_s^b$, can deduce whether a can also see e. Assuming all agents' binary masks are known can give rise to common knowledge about entities.

The *mutual knowledge* $\mathcal{M}_s^{\mathcal{G}}$ of a group of agents $\mathcal{G} \subseteq \mathcal{A}$ in state s is the set of entities that all agents in the group can see in that state: $\mathcal{M}_s^{\mathcal{G}} := \cap_{a \in \mathcal{G}} \mathcal{M}_s^a$. However, mutual knowledge does not imply common knowledge. Instead, the common knowledge $\mathcal{I}_s^{\mathcal{G}}$ of group $\mathcal{G}$ in state s is the set of entities such that all agents in $\mathcal{G}$ see $\mathcal{I}_s^{\mathcal{G}}$, know that all other agents in $\mathcal{G}$ see $\mathcal{I}_s^{\mathcal{G}}$, know that they know that all other agents see $\mathcal{I}_s^{\mathcal{G}}$, etc.

To know that another agent b also sees $e \in \mathcal{E}$, agent a must see b and b must see e, i.e., $\mu^a(s^a, s^b) \wedge \mu^b(s^b, s^e) = \top$. Common knowledge $\mathcal{I}_s^{\mathcal{G}}$ can then be formalised recursively for every agent $a \in \mathcal{G}$ as:

$$\mathcal{I}_s^{\mathcal{G}} \;=\; \lim_{m \to \infty} \mathcal{I}_s^{a,m}\,, \qquad \mathcal{I}_s^{a,0} \;=\; \mathcal{M}_s^a\,, \tag{1}$$

$$\mathcal{I}_s^{a,m} \;=\; \bigcap_{b \in \mathcal{G}} \left\{ e \in \mathcal{I}_s^{b,m-1} \,|\, \mu^a(s^a, s^b) \right\}. \tag{2}$$

This definition formalises the above description that common knowledge is the set of entities that a group member sees ($m = 0$), that it knows all other group members see as well ($m = 1$), and so forth ad infinitum.

The following lemma establishes that, in our setting, if a group of agents can all see each other, their common knowledge is their mutual knowledge.

Lemma 1. *In the setting described in the previous Section, and when all masks are known to all agents, the common knowledge of a group of agents $\mathcal{G}$ in state s is*

$$\mathcal{I}_s^{\mathcal{G}} \;=\; \begin{cases} \mathcal{M}_s^{\mathcal{G}} & \text{, if } \bigwedge\limits_{a,b \in \mathcal{G}} \mu^a(s^a, s^b) \\ \varnothing & \text{, otherwise} \end{cases}. \tag{3}$$

Proof. The lemma follows by induction on m. The recursive definition of common knowledge (2) holds trivially if $\mathcal{I}_s^{\mathcal{G}} = \varnothing$. Starting from the knowledge of any agent a in state s, $\mathcal{I}^{a,0} = \mathcal{M}_s^a$, definition (2) yields:

$$\mathcal{I}^{a,1} = \begin{cases} \mathcal{M}_s^{\mathcal{G}} & \text{, if } \bigwedge\limits_{b \in \mathcal{G}} \mu^a(s^a, s^b) \\ \varnothing & \text{, otherwise} \end{cases}.$$

Next we will show inductively that if all agents in group $\mathcal{G}$ know the mutual knowledge $\mathcal{M}_s^{\mathcal{G}}$ of state s at some iteration m, that is, $\mathcal{I}_s^{c,m} \overset{\text{ind.}}{=} \mathcal{M}_s^{\mathcal{G}}$, then this mutual knowledge becomes common knowledge 2 iterations later. Applying the definition (2) for any agent $a \in \mathcal{G}$ twice yields:

$$\begin{aligned} \mathcal{I}_s^{a,m+2} \;&=\; \left\{ e \in \mathcal{E} \,\middle|\, \bigwedge_{b \in \mathcal{G}} \left(\mu^a(s^a, s^b) \wedge \bigwedge_{c \in \mathcal{G}} \left(\mu^b(s^b, s^c) \wedge e \in \mathcal{I}_s^{c,m} \right) \right) \right\} \\ &\overset{\text{ind.}}{=}\; \left\{ e \in \mathcal{M}_s^{\mathcal{G}} \,\middle|\, \bigwedge_{b,c \in \mathcal{G}} \mu^b(s^b, s^c) \right\} \;=\; \mathcal{I}_s^{\mathcal{G}}\,, \end{aligned}$$

where we used

$$\bigwedge_{b \in \mathcal{G}} \left(\mu^a(s^a, s^b) \,\wedge\, \bigwedge_{c \in \mathcal{G}} \mu^b(s^b, s^c) \right) = \bigwedge_{b,c \in \mathcal{G}} \mu^b(s^b, s^c)\,,\ \forall a \in \mathcal{G}\,.$$

Therefore, starting at the knowledge any agent of group $\mathcal{G}$, in which all agents can see each other, the mutual knowledge becomes the common knowledge for all $m \geq 3$. □

The common knowledge can be computed using only the visible set $\mathcal{M}_s^a$ of every agent $a \in \mathcal{G}$. Moreover, actions that have been chosen by a policy, which itself is common knowledge, and that further depends only on common knowledge and a shared random seed can also be considered common knowledge. The common knowledge of group $\mathcal{G}$ up to time t is thus some common prior knowledge τ_0 and the commonly known trajectory $\tau_t^{\mathcal{G}} = (\tau_0, z_1^{\mathcal{G}}, \mathbf{u}_1^{\mathcal{G}}, \ldots, z_t^{\mathcal{G}}, \mathbf{u}_t^{\mathcal{G}})$, with $z_k^{\mathcal{G}} = \{s_k^e \,|\, e \in \mathcal{I}_{s_k}^{\mathcal{G}}\}$. Knowing all binary masks μ^a makes it possible to derive $\tau_t^{\mathcal{G}} = \mathcal{I}^{\mathcal{G}}(\tau_t^a)$ from the observation trajectory $\tau_t^a = (\tau_0, z_1^a, \ldots, z_t^a)$ of every agent $a \in \mathcal{G}$, and a function that conditions on $\tau^{\mathcal{G}}$ can therefore be computed independently by every member of $\mathcal{G}$. This idea of common knowledge based on local observations is illustrated in Figure 1.

Multi-Agent Common Knowledge Reinforcement Learning

The key idea behind MACKRL is to learn decentralised policies that are nonetheless coordinated. By conditioning the joint policy $\pi^{\mathcal{G}}(\mathbf{u}^{\mathcal{G}}_{\text{env}}|\mathcal{I}^{\mathcal{G}}(\tau^a))$ over joint action space $\mathcal{U}^{\text{env}}_{\mathcal{G}}$ of a group of agents $\mathcal{G}$ only on the common knowledge of that group, MACKRL learns centralized policies that can be executed in a decentralised fashion. Since all agents $a \in \mathcal{G}$ have access to $\mathcal{I}^{\mathcal{G}}(\tau^a)$ and the shared random seed, they can sample the same joint action $\mathbf{u}^{\mathcal{G}}_{\text{env}}$ and execute their individual component u^a_{env}.

If $\mathcal{I}^{\mathcal{G}}(\tau^a)$ is sufficiently informative, then in principle it is possible to learn a $\pi^{\mathcal{G}}$ that chooses high quality joint actions. However, if $\mathcal{I}^{\mathcal{G}}(\tau^a)$ is not sufficiently informative, it may be preferable for $\mathcal{G}$ to delegate action selection to smaller subgroups. Coordination would no longer occur across the subgroups, but action selection within each group could condition on a richer common knowledge signal. Furthermore, the choice between acting and delegating must itself be decentralisable amongst the agents in $\mathcal{G}$ and can thus only condition on $\mathcal{I}^{\mathcal{G}}(\tau^a)$.

MACKRL tackles this problem using a hierarchy of controllers. At the top level is a controller that can either select a joint action across all agents, $\mathbf{u}_{\text{env}}$, or specify a partition of the agents to delegate action selection to. At intermediate levels are controllers that can either select joint actions $\mathbf{u}^{\mathcal{G}}_{\text{env}}$ for a group $\mathcal{G}$ or a partition of $\mathcal{G}$. At the bottom level are controllers that select actions u^a_{env} for all individual agents that have not yet been assigned actions. Figure 2 illustrates the hierarchy for the pairwise version of this model discussed in the next section. Formally, the controller for a group $\mathcal{G}$ can be described as:

$$u^{\mathcal{G}} \sim \pi^{\mathcal{G}}\big(u^{\mathcal{G}} \,|\, \mathcal{I}^{\mathcal{G}}(\tau^a)\big) \qquad \text{with} \qquad u^{\mathcal{G}} \in \mathcal{U}_{\mathcal{G}}\,, \quad (4)$$

where $\mathcal{U}_{\mathcal{G}} = \{\mathcal{U}^{\text{env}}_{\mathcal{G}} \cup \mathcal{P}(\mathcal{G})\}$ and $\mathcal{P}(\mathcal{G})$ is the set of all partitions of $\mathcal{G}$. Algorithm 1 summarises MACKRL's hierarchical action selection. Since the hierarchical controllers condition only on the common knowledge as defined in (2), the actions can be executed by each agent. We use $\mathbf{u}^{\mathcal{P}} = \prod_{\mathcal{G}\in\mathcal{P}} u^{\mathcal{G}}$ to denote the joint action space of controllers in a partition $\mathcal{P}$.

Algorithm 1 Action Selection in MACKRL

```
function GETACTION(τ_t^a)
    Set u_env = 0                          ▷ Initialise joint action
    Set b = [A] ▷ Initialise group buffer with entire group
    while not b.empty() do
        Set G = b.pop()  ▷ Get the next group from buffer
        Sample u^G ~ π^G(u^G | I^G(τ_t^a))
        if u^G ∈ U^env_G then   ▷ Sample a joint action for G
            u_env[G] = u^G   ▷ Assign joint action elements
        else                          ▷ Delegate to selected partitions
            for G' ∈ u^G do
                b.push(G') ▷ Add groups G' in partition u^G
    return u_env
```

However, rather than training a set of factorised policies, one for each agent, we train the marginalised joint policy $p(\mathbf{u}_{\text{env}}|s)$ induced by the hierarchical sampling process:

$$p(\mathbf{u}_{\text{env}}|s) = \sum_{\text{path}\in\text{Paths}} p(\mathbf{u}_{\text{env}}|s, \text{path})p(\text{path}|s), \quad (5)$$

where Paths is the set of all possible action assignments of the hierarchical controllers, that is, each path is a possible outcome of the action-selection in Algorithm 1. In general, sampling from a joint probability is problematic since the number of logits grows exponentially in the number of agents. Furthermore, evaluating the joint probability requires central state information, and thus cannot be decentralised. The key insight of MACKRL is that we can use the hierarchical, decentralised process for sampling actions, while the marginal probabilities need to be computed only during training in order to obtain the joint probability of the joint action that was actually selected.

To train these policies, we use an actor-critic setup with a centralised baseline (Central-V, Foerster et al., 2018). In contrast to the decentralised policy, we condition the critic on the central state and the last actions of all agents. Since MACKRL induces a correlated probability across the joint action space, it effectively turns training into a single agent problem and renders the COMA baseline proposed in Foerster et al. (2018) non-applicable.

The gradient w.r.t. the policy parameters θ is:

$$\nabla_\theta J^p_t \quad = \quad A_t \; \nabla_\theta \log(p(\mathbf{u}_{\text{env},t}|s_t)), \quad (6)$$

where $A_t = (r_{t+1} + \gamma V(s_{t+1}, \mathbf{u}_{\text{env},t}) - V(s_t, \mathbf{u}_{\text{env},t-1}))$ is an estimate of the advantage. The value function is learned by gradient descend on the TD(λ) loss (Sutton and Barto, 1998). As in Central-V, all critics are trained on-policy using samples from the environment, and all controllers within one layer of the hierarchy share parameters.

In general, both the number of possible partitions and groups per partition can be large, making learning difficult. However, the next section describes an example implementation that illustrates how learning can be made tractable.

Pairwise MACKRL

In this paper, we focus on a simple implementation of MACKRL that we call *pairwise MACKRL*. The hierarchy consists of three levels. At the top level, the controller π_{ps} is a *pair selector* that is not allowed to directly select a joint action but can only choose among pairwise partitions, i.e., it must group the agents into disjoint pairs. If there are an odd number of agents, then one agent is put in a singleton group. At the second level, each controller $\pi^{aa'}_{\text{pc}}$ is a *pair controller* whose action space consists of the joint action space of the pair of agents plus a delegate action u_d that further partitions the pair into singletons: $\mathcal{U}_{\mathcal{G}} = \mathcal{U}^a_{\text{env}} \times \mathcal{U}^{a'}_{\text{env}} \cup \{u_d\}$, where $\mathcal{G} = \{a, a'\}$. At the third level, each controller selects an individual action u^a_{env} for a single agent a. Figure 2 illustrates pairwise MACKRL.

Experiments and Results

We evaluate MACKRL on two tasks. The first is a matrix game with partial observability. The state consists of two randomly sampled bits, which are drawn iid. The first bit indicates the *information state* and is always observable by both agents. The second bit selects which one of the two normal form games the agents are playing and is sampled 50%. When the first bit is in the 'common knowledge' state, which happens with a probability of $P(\text{common knowledge})$,

$$\begin{pmatrix} 1 & 0 & 0 & 0.4 & 0 \\ 0 & 0.2 & 0.4 & 0.8 & 0.4 \\ 0 & 0 & 0 & 0.4 & 0 \\ 0 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3: Matrix A

$$\begin{pmatrix} 0 & 0 & 0.2 & 0 & 1 \\ 0 & 0 & 0.4 & 0 & 0 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0 & 0 & 0.4 & 0 & 0 \\ 1 & 0 & 0.2 & 0 & 0 \end{pmatrix}$$

Figure 4: Matrix B

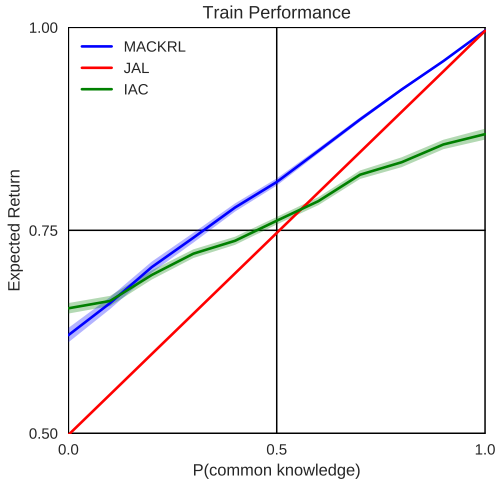

Figure 5: Results for the matrix game.

Level	Policy / Controller	#π
1	$\pi_{\text{ps}}(u^{\text{ps}}\|u^{\text{ps}}_{t-1}, h^{\text{ps}}_{t-1}, \mathcal{I}^{\mathcal{A}}_{s_t})$	1
2	$\pi^{aa'}_{\text{pc}}(u^{aa'}\|u^{aa'}_{t-1}, h^{aa'}_{t-1}, aa', \mathcal{I}^{aa'}_{s_t})$	3
3	$\pi^{a}(u^{a}\|o^{a}_{t}, h^{a}_{t-1}, u^{a}_{t-1}, a)$	3

Table 1: Hierarchy of pairwise MACKRL, where h is the hidden state of RNNs and o^a_t are observations. #π shows the number of controllers on this level for 3 agents.

the matrix bit is always observable by both agents and is thus common knowledge.

In contrast, when the first bit is in the 'obfuscation state', each of the agents observes the 'matrix bit' with a probability of 50% (iid), indicating whether the reward will be chosen using the matrix shown in Figure 3 or the matrix shown in Figure 4, and receives a 'no observation' otherwise. The code including a proof of principle implementation is available in the Supplementary Material and published online. We explore how MACKRL performs, for a range of values for $P(\text{common knowledge})$, in comparison to an always centralised policy using only the common knowledge and fully decentralised learners.

Figure 5 shows performance on the matrix game as a function of the probability of the common knowledge bit. When there is no common knowledge MACKRL reproduces the performance of independent actors (IAC), while a centralised policy, learned with *joint-action-learning* (JAL) (Busoniu, Babuska, and De Schutter, 2008), fails to take advantage of the private observations. When common knowledge is always available MACKRL matches the performance of JAL. For intermediate probabilities, MACKRL outperforms both IAC and JAL. Inspecting the policy verifies that MACKRL learns to delegate to the independent learners whenever the common knowledge bit is absent but carries out a joint action when the bit is present. This corresponds to the optimal policy.

The second task is a challenging multi-agent version of StarCraft II micromanagement. Each unit is controlled by a decentralised agent and has to rely on local information for action selection. We use the multi-agent version of the game as described in (Rashid et al., 2018) and use the maps where both sides have either 3 Marines (denoted '3m') or 2 Stalkers and 3 Zealots (denoted '2s3z'). The '3m' map was chosen as a simple reference map since it has been tackled in previous work. The '2s3z' consists of different unit types with different attack abilities and thus allows for more complex coordination between the agents. In particular Rashid et al. (2018) showed that independent learners fail to perform well on the '2s3z' map, making it a good testbed for our method.

All policies are implemented as 2-layer recurrent neural networks (GRUs) with 64 hidden units, while the critic is feed-forward and uses full state information. Table 1 describes the inputs and action spaces at each level of the hierarchy. Parameters are shared across controllers within each of the second and third levels of the hierarchy. Therefore, we also feed in the agent index or index pairs into the policy. For exploration we used a bounded softmax distribution in which the agent samples from a softmax over the policy logits with probability $(1-\epsilon)$ and samples randomly with probability ϵ. ϵ was annealed from 0.5 to the final value of 0.01 across the first 50k environment steps.

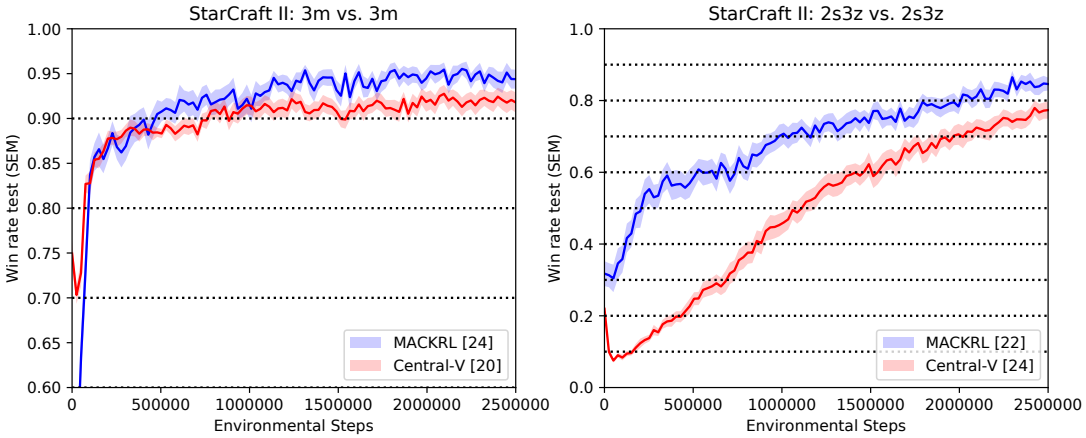

Figure 6: Mean and standard error of the mean of test win rates for two levels in StarCraft II: one with 5 marines (left), and one with 2 stalkers and 3 zealots on each side (right). Also shown is the number of runs [in brackets].

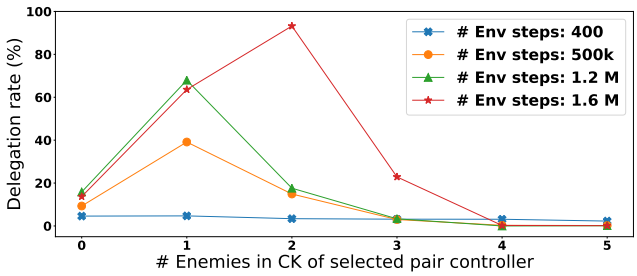

Figure 7: Delegation rate vs. number of enemies (2s3z) in the common knowledge of the pair controller over training.

Episodes were collected using 8 parallel environment of SCII, and optimisation was done on GPU using ADAM with a learning rate of 0.0005 for both the agents and the critic. The policies were fully unrolled and updated in a large minibatch of $T \times B$ entries, where $T = 60$ and $B = 8$. In contrast the critic was optimised in small minibatches of size 8, one for each timestep, looping backwards in time. We found that this both stabilised but also accelerated training compared to doing full batch updates for the critic. The target network for the critic was updated after every 200 update steps for the critic. We also used a TD-lambda of 0.8 to accelerate reward propagation across time.

Figure 6 shows the win rate of StarCraftII agents during test trajectories on our two maps. We omit independent learning since it is known to do poorly in this setting (Rashid et al., 2018). On the easier map (3m), both methods achieve near state-of-the-art performance above 90%. Since Central-V has around three times fewer parameters, it is able to initially learn faster on this simple map, but MACKRL achieves marginally higher final performance. On the more challenging map with mixed unit types (2s3z), MACKRL learns faster and achieves a higher final score. These results show that MACKRL can outperform a fully factored policy when all other aspects of training are the same.

To demonstrate that the pair controller can indeed learn to delegate strategically, we plot in Figure 7 the percentage of delegation actions u_d against the number of enemies in the common knowledge of the selected pair controller, in situations where there is at least some common knowledge. Since we start with randomly initialised policies, at the beginning of training the pair controller delegates only rarely to the decentralised controllers. As training proceeds, it learns to delegate in most situations where the number of enemies in the common knowledge of the pair is small, the exception being no visible enemies, which happens too rarely (5% of cases). This shows that MACKRL can learn to delegate in order to take advantage of the private observations of the agents, but also learns to coordinate in the joint action space when there is substantial common knowledge.

Conclusion and Future Work

We introduce MACKRL, a method which allows a team of agents to learn a policy in the joint action space using common knowledge, such that the policy can still be executed in a fully decentralised fashion. MACKRL shows strong performance on a matrix game where it clearly beats a joint-action-learner and independent learners. We also test MACKRL on a challenging multi-agent version of StarCraftII micromanagement and find that it strongly outperforms a strong baseline in the harder of the two maps, and marginally on the easier one. In future work, we will investigate scaling MACKRL to settings with many agents and develop methods that allow agents to learn to deduce common knowledge from their trajectories, rather than having it presented separately.

Acknowledgements

We would like to thank Chia-Man Hung, Tim Rudner and Tabish Rashid for valuable discussions.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement number 637713), the National Institutes of Health (grant agreement number R01GM114311) and EPSRC/MURI grant EP/N019474/1. This work is linked to and partly funded by the project Free the Drones (FreeD) under the Innovation Fund Denmark and Microsoft. It was also supported by the Oxford-Google DeepMind Graduate Scholarship and a generous equipment grant from NVIDIA.

References

Albrecht, S. V., and Stone, P. 2017. Reasoning about hypothetical agent behaviours and their parameters. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 547–555. International Foundation for Autonomous Agents and Multiagent Systems.

Amato, C.; Konidaris, G. D.; and Kaelbling, L. P. 2014. Planning with macro-actions in decentralized pomdps. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, 1273–1280. International Foundation for Autonomous Agents and Multiagent Systems.

Aumann, R. J., et al. 1974. Subjectivity and correlation in randomized strategies. *Journal of mathematical Economics* 1(1):67–96.

Barrett, S.; Stone, P.; and Kraus, S. 2011. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 567–574. International Foundation for Autonomous Agents and Multiagent Systems.

Boutilier, C. 1999. Sequential optimality and coordination in multiagent systems. In *IJCAI*, volume 99, 478–485.

Brafman, R. I., and Tennenholtz, M. 2003. Learning to coordinate efficiently: A model-based approach. In *Journal of Artificial Intelligence Research*, volume 19, 11–23.

Busoniu, L.; Babuska, R.; and De Schutter, B. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews* 38(2):156.

Cao, Y.; Yu, W.; Ren, W.; and Chen, G. 2013. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics* 9(1):427–438.

Cigler, L., and Faltings, B. 2013. Decentralized anti-coordination through multi-agent learning. *Journal of Artificial Intelligence Research* 47:441–473.

Dietterich, T. G. 2000. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Int. Res.* 13(1):227–303.

Foerster, J.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2137–2145.

Foerster, J.; Nardelli, N.; Farquhar, G.; Torr, P.; Kohli, P.; Whiteson, S.; et al. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of The 34th International Conference on Machine Learning*.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual multi-agent policy gradients. In *AAAI*.

Genter, K.; Agmon, N.; and Stone, P. 2013. Ad hoc teamwork for leading a flock. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 531–538. International Foundation for Autonomous Agents and Multiagent Systems.

Genter, K.; Laue, T.; and Stone, P. 2017. Three years of the robocup standard platform league drop-in player competition. *Autonomous Agents and Multi-Agent Systems* 31(4):790–820.

Guestrin, C.; Koller, D.; and Parr, R. 2002. Multiagent planning with factored mdps. In *Advances in neural information processing systems*, 1523–1530.

Gupta, J. K.; Egorov, M.; and Kochenderfer, M. 2017. Cooperative multi-agent control using deep reinforcement learning.

Jorge, E.; Kageback, M.; and Gustavsson, E. 2016. Learning to play guess who? and inventing a grounded language as a consequence. *arXiv preprint arXiv:1611.03218*.

Kok, J. R., and Vlassis, N. 2004. Sparse cooperative q-learning. In *Proceedings of the twenty-first international conference on Machine learning*, 61. ACM.

Korkmaz, G.; Kuhlman, C. J.; Marathe, A.; Marathe, M. V.; and Vega-Redondo, F. 2014. Collective action through common knowledge using a facebook model. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, 253–260. International Foundation for Autonomous Agents and Multiagent Systems.

Kraemer, L., and Banerjee, B. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190:82–94.

Kumar, S.; Shah, P.; Hakkani-Tur, D.; and Heck, L. 2017. Federated control with hierarchical multi-agent deep reinforcement learning. *arXiv preprint arXiv:1712.08266*.

Liu, M.; Sivakumar, K.; Omidshafiei, S.; Amato, C.; and How, J. P. 2017. Learning for multi-robot cooperation in partially observable stochastic environments with macro-actions. *arXiv preprint arXiv:1707.07399*.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*.

Makar, R.; Mahadevan, S.; and Ghavamzadeh, M. 2001. Hierarchical multi-agent reinforcement learning. In *Proceedings of the fifth international conference on Autonomous agents*, 246–253. ACM.

Makino, T., and Aihara, K. 2006. Multi-agent reinforcement learning algorithm to handle beliefs of other agents' policies and embedded beliefs. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 789–791. ACM.

Oliehoek, F. A.; Spaan, M. T. J.; and Vlassis, N. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. 32:289–353.

Omidshafiei, S.; Pazis, J.; Amato, C.; How, J. P.; and Vian, J. 2017. Deep decentralized multi-task multi-agent rl under partial observability. *arXiv preprint arXiv:1703.06182*.

Osborne, M. J., and Rubinstein, A. 1994. *A course in game theory*. MIT press.

Panella, A., and Gmytrasiewicz, P. 2017. Interactive pomdps with finite-state models of other agents. *Autonomous Agents and Multi-Agent Systems* 31(4):861–904.

Peng, P.; Yuan, Q.; Wen, Y.; Yang, Y.; Tang, Z.; Long, H.; and Wang, J. 2017. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*.

Rashid, T.; Samvelyan, M.; de Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of The 35th International Conference on Machine Learning*.

Rubinstein, A. 1989. The electronic mail game: Strategic behavior under" almost common knowledge". *The American Economic Review* 385–391.

Sukhbaatar, S.; Szlam, A.; and Fergus, R. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, 2244–2252.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Synnaeve, G.; Nardelli, N.; Auvolat, A.; Chintala, S.; Lacroix, T.; Lin, Z.; Richoux, F.; and Usunier, N. 2016. Torchcraft: a library for machine learning research on real-time strategy games. *arXiv preprint arXiv:1611.00625*.

Tan, M. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, 330–337.

Thomas, K. A.; DeScioli, P.; Haque, O. S.; and Pinker, S. 2014. The psychology of coordination and common knowledge. *Journal of personality and social psychology* 107(4):657.

Usunier, N.; Synnaeve, G.; Lin, Z.; and Chintala, S. 2016. Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. *arXiv preprint arXiv:1609.02993*.

Yang, E., and Gu, D. 2004. Multiagent reinforcement learning for multi-robot systems: A survey. Technical report, tech. rep.