Cooperative Multi-agent Control Using Deep Reinforcement Learning

Jayesh K. Gupta(✉), Maxim Egorov, and Mykel Kochenderfer

Stanford University, Stanford, USA
jkg@cs.stanford.edu, {megorov,mykel}@stanford.edu

Abstract. This work considers the problem of learning cooperative policies in complex, partially observable domains without explicit communication. We extend three classes of single-agent deep reinforcement learning algorithms based on policy gradient, temporal-difference error, and actor-critic methods to cooperative multi-agent systems. To effectively scale these algorithms beyond a trivial number of agents, we combine them with a multi-agent variant of curriculum learning. The algorithms are benchmarked on a suite of cooperative control tasks, including tasks with discrete and continuous actions, as well as tasks with dozens of cooperating agents. We report the performance of the algorithms using different neural architectures, training procedures, and reward structures. We show that policy gradient methods tend to outperform both temporal-difference and actor-critic methods and that curriculum learning is vital to scaling reinforcement learning algorithms in complex multi-agent domains.

1 Introduction

Cooperation between several interacting agents has been well studied [1–3]. While the problem of cooperation can be formulated as a decentralized partially observable Markov decision process (Dec-POMDP), exact solutions are intractable [4,5]. A number of approximation methods for solving Dec-POMDPs have been developed recently that adapt techniques ranging from reinforcement learning [6] to stochastic search [7]. However, applying these methods to real-world problems is challenging because they are typically limited to discrete action spaces and require carefully designed features.

On the other hand, recent work in single agent reinforcement learning has enabled learning in domains that were previously thought to be too challenging due to their large and complex observation spaces. This line of work combines ideas from deep learning with earlier work on function approximation [8,9], giving rise to the field of deep reinforcement learning. Deep reinforcement learning has been successfully applied to complex real-world tasks that range from playing Atari games [10] to robotic locomotion [11]. The recent success of the field leads to a natural question—how well can ideas from deep reinforcement learning be applied to cooperative multi-agent systems?

© Springer International Publishing AG 2017
G. Sukthankar and J. A. Rodriguez-Aguilar (Eds.): AAMAS 2017 Best Papers,
LNAI 10642, pp. 66–83, 2017.
https://doi.org/10.1007/978-3-319-71682-4_5

In this work, we focus on problems that can be modeled as Dec-POMDPs. We extend three classes of deep reinforcement learning algorithms: temporal-difference learning using Deep Q Networks (DQN) [10], policy gradient using Trust Region Policy Optimization (TRPO) [12], and actor-critic using Deep Deterministic Policy Gradients (DDPG) [13] and A3C [14]. We consider three training schemes for multi-agent systems based on centralized training and execution, concurrent training with decentralized execution, and parameter sharing during training with decentralized execution. We incorporate curriculum learning [15] into cooperative domains by first learning policies that require a small number of cooperating agents and then gradually increasing the number of agents that need to cooperate. The algorithms and training schemes are benchmarked on four multi-agent tasks requiring cooperative behavior. The benchmark tasks were chosen to represent a diverse variety of complex environments with discrete and continuous actions and observations.

Our empirical evaluations show that multi-agent policies trained with parameter sharing and an appropriate choice of reward function exhibit cooperative behavior without explicit communication between agents. We show that the multi-agent extension of TRPO outperforms all other algorithms on benchmark problems with continuous action spaces, while A3C has the best performance on the discrete action space benchmark. By combing curriculum learning and TRPO, we demonstrate scalability of deep reinforcement learning in large, continuous action domains with dozens of cooperating agents and hundreds of agents present in the environment. To our knowledge, this work presents the first cooperative reinforcement learning algorithm that can successfully scale in large continuous action spaces. The benchmark problems and the implementations of multi-agent algorithms can be found at https://github.com/sisl/MADRL.

2 Related Work

Multi-agent reinforcement learning has a rich literature [2,16]. A number of algorithms involve value function based cooperative learning. Tan compared the performance of cooperative agents to independent agents in reinforcement learning settings [1]. Ono and Fukumoto identified modularity as a useful prior to simplify the application of reinforcement learning methods to multiple agents [17]. Guestrin et al. later extended this idea and factored the joint value function into a linear combination of local value functions and used message passing to find the joint optimal actions [18]. Lauer and Riedmiller tried distributing the value function into learning multiple tables but failed to scale to stochastic environments [19].

Policy search methods have found better success in partially observable environments [20]. Peshkin et al. studied gradient based distributed policy search methods [21]. Our solution approach can be considered a direct descendant of the techniques introduced in their work. However, instead of using finite state machines, our model uses deep neural networks to control the agents. This approach allows us to extend neural network controllers to tasks with continuous actions, use deep reinforcement learning optimization techniques, and consider more complex observation spaces.

Relatively little work on multi-agent reinforcement learning has focused on continuous action domains. A few notable approaches include those of Fernández and Parker who focus on discretization and Tamakoshi and Ishii who used a normalized Gaussian Network as a function approximator to learn continuous action policies [22,23]. Many of these approaches only work in fairly restricted settings and fail to scale to high-dimensional raw observations or continuous actions. Moreover, their computational complexity grows exponentially with the number of agents.

Multi-agent control has also been studied in extensive detail from the dynamical systems perspective in problems like formation control [24], coverage control [25], and consensus [26]. The limitations of the dynamical systems approach lie in its requirement for hand-engineered control laws and problem specific features. While the approach allows for development of provable characteristics about the controller, it requires extensive domain knowledge and hand engineering. Overall, deep reinforcement learning provides a more general way to solve multi-agent problems without the need for hand-crafted features and heuristics by allowing the neural network to learn those properties of the controller directly from raw observations and reward signals.

Recent research has applied deep reinforcement learning to multi-agent problems. Tampuu et al. extended the DQN framework to independently train multiple agents [27]. Specifically, they demonstrate how collaborative and competitive behavior can arise with the appropriate choice of reward structure in a two-player Pong game. More recently, Foerster et al. and Sukhbaatar et al. train multiple agents to learn a communication protocol to solve tasks with shared utility [28,29]. They demonstrate end-to-end differentiable training using novel neural architectures. However, these examples work with either relatively few agents or simple observations and do not share our focus on decentralized control problems with high-dimensional observations and continuous action spaces.

3 Background

In this work, we consider multi-agent domains that are fully cooperative and partially observable. All agents are attempting to maximize the discounted sum of joint rewards. No single agent can observe the state of the environment. Instead, each agent receives a private observation that is correlated with that state. We assume the agents cannot explicitly communicate and must learn cooperative behavior only from their observations.

Formally, the problems considered in this work can be modeled as Dec-POMDPs defined by the tuple $(\mathcal{I}, \mathcal{S}, \{\mathcal{A}_i\}, \{\mathcal{Z}_i\}, T, R, O)$, where $\mathcal{I}$ is a finite set of agents, $\mathcal{S}$ is a set of states, $\{\mathcal{A}_i\}$ is a set of actions for each agent i, $\{\mathcal{Z}_i\}$ is a set of observations for each agent i, and T, R, O are the joint transition, reward, and observation models, respectively. In this work, we consider problems where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{Z}$ can be infinite to account for continuous domains. In the reinforcement learning setting, we do not know T, R, or O, but instead have access to a generative model. It is natural to also consider a centralized model known as a multi-agent POMDP (MPOMDP), with joint action and observation models. The centralized nature of MPOMDPs makes them less effective at scaling to systems with many agents.

In the reminder of the section, we briefly describe four single-agent deep reinforcement learning algorithms, including temporal-difference, actor-critic, and policy gradient approaches. We also discuss the roles of reward shaping and curriculum learning in multi-agent settings.

3.1 Deep Q-Network

The DQN algorithm [10] is a temporal-difference method that uses a neural network to approximate the state-action value function. DQN relies on an experience replay dataset $\mathcal{D}_t = \{e_1, \ldots, e_t\}$, which stores the agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ to reduce correlations between observations. The experience consists of the current state s_t, the action the agent took a_t, the reward it received r_t, and the state it transitioned to s_{t+1}. The learning update at each iteration i uses a loss function based on the temporal-difference update:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i) \right)^2 \right]$$

where θ_i and θ_i^- are the parameters of the Q-networks and a target network respectively at iteration i, and the experience samples (s, a, r, s') are sampled uniformly from $\mathcal{D}$. In partially observable domains where only observations o_t are available at time t instead of the entire state s_t, the experience takes the form $e_t = (o_t, a_t, r_t, o_{t+1})$. One of the limitations of DQN is that it cannot easily handle continuous action spaces.

3.2 Deep Deterministic Policy Gradient

DDPG combines the actor-critic and DQN approaches to learn policies in domains with continuous actions. DDPG maintains a parameterized actor function $\mu(s \mid \theta^\mu)$, which deterministically maps states to actions while learning a critic $Q(s, a)$ that estimates the value of state-action pairs. The actor can be updated with the following optimization step:

$$\nabla_{\theta^\mu} J \approx \mathbb{E}_{s_t \sim \rho_\pi} [\nabla_a Q(s, a \mid \theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s \mid \theta^\mu)|_{s=s_t}]$$

where ρ_π are transitions generated from a stochastic behavior policy π, typically represented with a Gaussian distribution centered at $\mu(s \mid \theta^\mu)$.

3.3 Asynchronous Advantage Actor Critic

Asynchronous Advantage Actor Critic (A3C) [14] consists of global shared networks for policy $\pi(a \mid s, \theta_p)$ and value $V(s, \theta_v)$ functions. Multiple copies running independently accumulate gradients in parallel to asynchronously update this network. The policy gradients are given by:

$$\nabla_{\theta_p} \log \pi(a_t \mid s_t; \theta_p) A(s_t, a_t; \theta_v)$$

where the advantage function $A(s_t, a_t; \theta_v)$ is computed from difference between returns from n-step rollout and value function output.

The value network loss function is to minimize squared error of value function outputs from environment returns.

3.4 Trust Region Policy Optimization

TRPO [12] is a policy gradient method that allows precise control of the expected policy improvement during the optimization step. At each iteration k, TRPO aims to solve the following constrained optimization problem by optimizing the stochastic policy π_θ:

$$\begin{aligned} \underset{\theta}{\text{Maximize}} \quad & \mathbb{E}_{s \sim \rho_{\theta_k}, a \sim \pi_{\theta_k}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_{\theta_k}(s, a) \right] \\ \text{subject to} \quad & \mathbb{E}_{s \sim \rho_{\theta_k}} \left[D_{KL}(\pi_{\theta_k}(\cdot|s) \| \pi_\theta(\cdot|s)) \right] \leq \Delta_{KL} \end{aligned}$$

where $\rho_\theta = \rho_{\pi_\theta}$ are the discounted state-visitation frequencies induced by π_θ. $A_{\theta_k}(s, a)$ is the advantage function, which can be estimated by the difference between the empirical returns and the baseline. We use a linear value function baseline in our experiments. D_{KL} is the KL divergence between the two policy distributions, and Δ_{KL} is a step size parameter that controls the maximum change in policy per optimization step. The expectations in the expression can be evaluated using sample averages, and the policy can be represented by non-linear function approximators such as neural networks. The stochastic policy π_θ can be represented by a categorical distribution when the actions of the agent are discrete and by a Gaussian distribution when the actions are continuous.

3.5 Reward Structure

The concept of reward shaping [30] involves modifying rewards to accelerate learning without changing the optimal policy. When modeling a multi-agent system as a Dec-POMDP, rewards are shared jointly by all agents. In a centralized representation, the reward signal cannot be decomposed into separate components, and is equivalent to the joint reward in a Dec-POMDP. However, decentralized representations allow us an alternative local reward representation. Local rewards can restrict the reward signal to only those agents that are involved in the success or failure at a task. Bagnell and Ng have shown that such local information can help reduce the number of samples required for learning [31]. As we will note later, this decomposition can drastically improve training time. The performance of the policy is still evaluated using the global reward.

3.6 Curriculum Learning

Curriculum learning leverages the idea of learning policies for simple tasks first, and then building on that knowledge to solve more difficult tasks [15]. Formally, a curriculum $\mathcal{T}$ is an ordered set of tasks organized by increasing difficulty. In cooperative settings, the tasks in the curriculum become more difficult as the number of cooperating agents required to complete the task increases.

4 Cooperative Reinforcement Learning

This section outlines three training schemes for multi-agent reinforcement learning in cooperative settings as well as their advantages and disadvantages.

4.1 Centralized

A centralized policy maps the joint observation of all agents to a joint action, and is equivalent to a MPOMDP policy. A major drawback of this approach is that it is centralized in both training and execution, and leads to an exponential growth in the observation and actions spaces with the number of agents. We address this intractability in part by factoring the action space of centralized multi-agent systems.

We first assume that the joint action can be factored into individual components for each agent. The factored centralized controller can then be represented as a set of sub-policies that map the joint observation to an action for a single agent. In the policy gradient approach this reduces to factoring the joint action probability as $P(\boldsymbol{a}) = \prod_i P(a_i)$ where a_i are the individual actions of an agent. In practice, this means that the policy of a given agent is represented by a subset of the output nodes in the neural network. In systems with discrete actions, this reduces the size of the action space from $|\mathcal{A}|^n$ to $n|\mathcal{A}|$, where n is the number of agents and $\mathcal{A}$ is the action space for a single agent (we assume homogeneous agents for simplicity). While this is a significant reduction in the size of the action space, the exponential growth in the observation spaces ultimately makes centralized controllers impractical for complex cooperative tasks.

4.2 Concurrent

In concurrent learning, each agent learns its own individual policy. Concurrent policies map an agent's private observation to an action for that agent. Each agent's policy is independent. In the policy gradient approach, this means optimizing multiple policies simultaneously from the joint reward signal. One of the advantages of this approach is that it makes learning of heterogeneous policies easier. This can be beneficial in domains where agents may need to take on specific roles in order to coordinate and receive reward.

The major drawback of concurrent training is that it does not scale well to large numbers of agents. Because the agents do not share experience with one

Algorithm 1. PS-TRPO

```
Input: Initial policy parameters $\Theta_0$, trust region size $\Delta$
for $i \leftarrow 0, 1, \ldots$ do
    Rollout trajectories for all agents $\boldsymbol{\tau} \sim \pi_{\theta_i}$
    Compute advantage values $A_{\pi_{\theta_i}}(o^m, m, a^m)$ for each agent $m$'s trajectory element.
    Find $\pi_{\theta_{i+1}}$ maximizing Eq. (1)
        subject to $\overline{D}_{KL}(\pi_{\theta_i} \| \pi_{\theta_{i+1}}) \leq \Delta$
```

another, this approach adds additional sample complexity to the reinforcement learning task. Another drawback of the approach is that the agents are learning and adjusting their policies individually making the environment dynamics non-stationary, which can lead to instability.

4.3 Parameter Sharing

The policies of homogeneous agents may be trained more efficiently using parameter sharing. This approach allows the policy to be trained with the experiences of all agents simultaneously. However, it still allows different behavior between agents because each agent receives unique observations, which includes their respective index. In parameter sharing, the control is decentralized but the learning is not. In the remainder of the paper, all training schemes use parameter sharing unless stated otherwise.

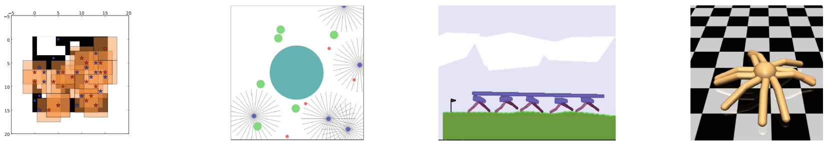
So long as the agents can execute decentralized policies with shared parameters, single agent algorithms like DDPG, DQN, TRPO and A3C can be extended to multi-agent systems. As an example, Algorithm 1 describes a policy gradient approach that combines parameter sharing and TRPO. We refer to it as PS-TRPO. We first initialize the policy network and set the step size parameter. At each iteration of the algorithm, the policy with shared parameters is used by each agent to generate trajectories. The batch of trajectories from all the agents is used to compute the advantage value and maximize the following objective:

$$L(\theta) = \mathbb{E}_{o \sim \rho_{\theta_k}, a \sim \pi_{\theta_k}} \left[\frac{\pi_\theta(a \mid o, m)}{\pi_{\theta_k}(a \mid o, m)} A_{\theta_k}(o, m, a) \right] \tag{1}$$

where m is the agent index. The results of the optimization are used to compute the parameter update for the policy.

5 Tasks

The four multi-agent benchmark tasks are described in this section. All tasks are partially observable. For more details we refer the reader to the source code.



(a) Pursuit (b) Waterworld (c) Multi-Walker (d) Multi-Ant

Fig. 1. Examples of the four cooperative domains. (Color figure online)

5.1 Discrete

Pursuit. Pursuit is a standard task for benchmarking multi-agent algorithms [32]. The pursuit-evasion domain consists of two sets of agents: evaders and pursuers. The evaders are trying to avoid pursuers, while the pursuers are trying to catch the evaders. The action and observation spaces in this problem are discrete. Each pursuer receives a range-limited observation of its surroundings, and must choose between five actions Stay, Go East, Go West, Go South, Go North. The observations contain information about the agent's surroundings, including the location of nearby pursuers, evaders, and obstacles. The example in Fig. 1a shows a 32×32 grid world with randomly generated obstacles, 20 pursuers (denoted by red stars), and 20 evaders (denoted by blue stars). The square box surrounding the pursuers indicates their observation range. The pursuers receive a reward of 5.0 when they surround and catch an evader, and a reward of 0.01 when they occupy the same space as an evader.

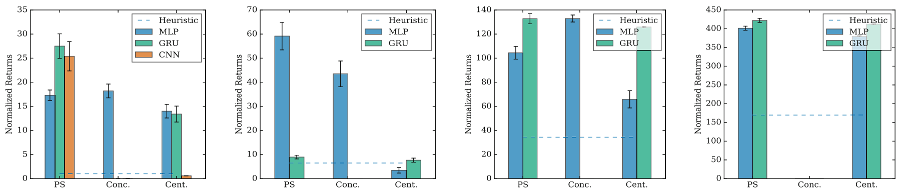
5.2 Continuous

Waterworld. Waterworld can be seen as an extension of the above mentioned pursuit problem to a continuous domain. The extension is based on the single agent waterworld domain used by [33]. In this task, agents need to cooperate to capture moving food targets while avoiding poison targets. Both the observation and action spaces are continuous, and the agents move around by applying a two-dimensional force. The agents receive a reward of 10.0 for capturing a food target, a reward of -1.0 for capturing a poison target, and an exertion penalty of $-0.01 \cdot \|a_i\|^2$.

Multi-Walker. Multi-Walker is a more difficult continuous control locomotion task based on the BipedalWalker environment from OpenAI gym [34]. The domain consists of multiple bipedal walkers that can actuate the joints in each of their legs. At the start of each simulation, a large package that stretches across all walkers is placed on top of the walkers. The walkers must learn how to move forward and to coordinate with other agents in order to keep the package balanced while navigating a complex terrain. Each agent receives a reward of 1.0 for moving the package forward 1 meter, a reward of -100.0 for falling, and a reward of -100.0 for dropping the package. An example environment with five walkers is shown in Fig. 1c.

Table 1. Summary of network architectures for each algorithm

	TRPO	DDPG/DQN	A3C
Feature Net	100-50-25	400-300	128
Recurrent	GRU-32	NA	LSTM-128
Activation	tanh	ReLU	tanh



(a) Pursuit (b) Waterworld (c) Multi-Walker (d) Multi-Ant

Fig. 2. Normalized average returns for multi-agent policies trained using TRPO. Missing entries indicate the training was unsuccessful. A random policy has zero normalized average return. Error bars represent standard error. The Wilcoxon test suggests the differences are significant ($p < 0.05$) except for the difference between centralized GRU and shared parameter GRU for the waterworld domain.

Multi-Ant. The multi-ant domain is a 3D locomotion task based on the quadrupedal robot used in [35]. The goal of the robot is to move forward as quickly as possible. In this domain, each leg of the ant is treated as a separate agent that is able to sense its own position and velocity as well as those of its two neighbors. Each leg is controlled by applying torque to its two joints. An example multi-ant with ten legs is shown in Fig. 1d.

6 Experiments

This section presents empirical results that compare the performance of multi-agent extensions of TRPO, DDPG, A3C, and DQN. In continuous action domains we compare TRPO, A3C, and DDPG, while in discrete action domains we compare TRPO, A3C, and DQN. We examine both feed-forward and recurrent policies in this work. We also examine the effects of centralized, concurrent, and shared parameters training schemes as well as two reward mechanisms that are relevant to multi-agent domains. The results are compared against each other and against a heuristic hand-crafted baseline for each task. Lastly, we demonstrate the benefits of curriculum learning to scalability in cooperative domains.

The neural network architectures used in this work are summarized in Table 1. The feature net represents the number of neurons in each layer and is used as the feedforward multi-layer perceptron (MLP) policy in each algorithm. The type of the hidden cell, either GRU or LSTM, and their number is indicated for recurrent policies. The feature net serves as the observation embedding for

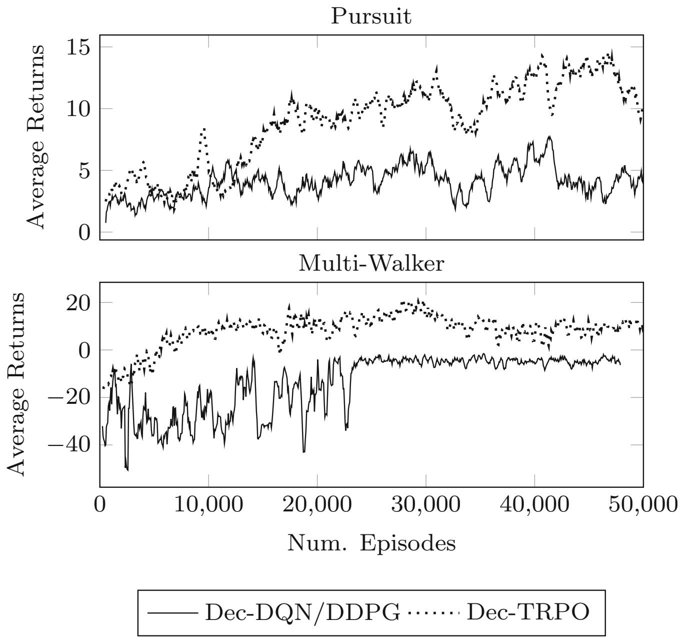


Fig. 3. Training curves comparing PS-TRPO and PS-DQN in Pursuit and PS-DDPG in Multi-Walker Domains.

recurrent policies. DQN/DDPG do not use recurrent policies, and A3C uses a single hidden layer as a feature network.

In all experiments, we use the discount factor $\gamma = 0.99$. For PS-TRPO, we set the step size to $\Delta = 0.01$, and constrain the size of each batch to a maximum of 24000 time-steps. For DDPG and DQN, we used batch sizes of 32, learning rate of 1×10^{-3} for the state-action value function and 1×10^{-4} for the policy network. For A3C, we used RMSProp [36] with an annealed learning rate starting from 5×10^{-5} with decay of 0.99.

6.1 Discrete Control Task

We first compared performances of the three training schemes on the pursuit problem using TRPO. The emergent behavior observed in TRPO policies included pursuers breaking up into teams to maximize the number of evaders that were captured. The results are summarized in Fig. 2a for a 16×16 grid, 8 pursuers with an observation range of 7, and 30 evaders. The figure shows that parameter sharing tends to outperform both the concurrent and centralized training schemes. Because the observation is image-like with spatial correlations present in each observation dimension, we also used a convolutional neural networks (CNN) to represent the policy in this task. The results show that with

Table 2. Average returns for parameter sharing multi-agent policies with global and local rewards

	Global	Local
Pursuit	8.1	12.1
Waterworld	−1.4	14.3
Multi-Walker	−23.3	29.9
Multi-Ant	475.2	488.1

parameter sharing, CNN policies outperform MLP policies, while GRU policies have the best overall performance.

We then compared the training behavior of global and local rewards. We found that using local rewards consistently improved convergence during training. An example of this difference for the pursuit evasion problem is shown in Table 2.

We compared the performance of PS-DQN against PS-TRPO and PS-A3C. As can be seen from Fig. 3 and Table 4, PS-A3C outperforms both PT-TRPO and PS-DQN, with PS-DQN having the worst performance. We hypothesize that PS-DQN is unable to learn a good controller due to the changing policies of other agents in the environment. This makes the dynamics of the problem non-stationary which causes experience replay to inaccurately describe the current state of the environment.

We also tested the ability of PS-TRPO to scale with very large observation spaces. The pursuit domain was set up on a 128×128 grid with 200 pursuers and 200 evaders with at least 16 pursuers required to capture an evader. While hundreds of agents are present in the environment, only 16 of them need to cooperate to achieve the capture task. Each observation is a four channel 21×21 image, making the observation space 1764 dimensional. The training curves for this task are shown in Fig. 4, and show that the MLP policy fails to learn a policy that can outperform the heuristic. However, by leveraging CNNs, we are able to outperform the heuristic in this complex domain.

Comparison to Traditional Method. Traditional reinforcement learning and Dec-POMDP approaches have difficulty solving problems with continuous action spaces and scale to problems with large numbers of agents. We also confirmed that PS-TRPO performs as well as a traditional approach for solving PS-TRPO on a small 5×5 grid pursuit problem. The approach we use as comparison resembles Joint Equilibrium search for policies (JESP) [37] in that it finds a policy that maximizes the joint expected reward for one agent at a time, while keeping the policies of all the other agents fixed. The process is repeated until an equilibrium is reached. In our approach, we use the fast informed bound (FIB) algorithm [38] to perform the policy optimization of a single agent.

The pursuit problem is set on a 5×5 grid with a square obstruction in the middle. There is a single evader and two pursuers. Both of the pursuers must

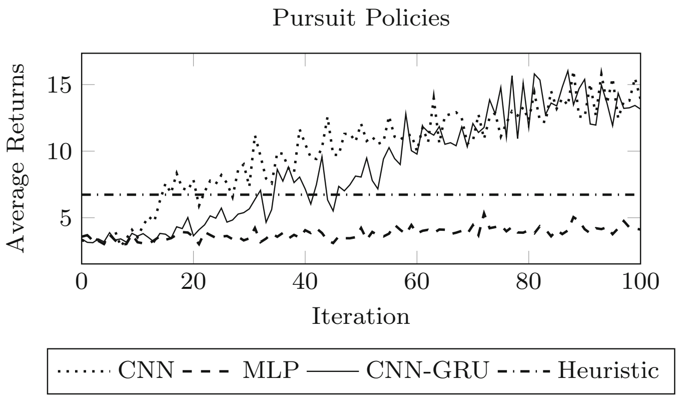


Fig. 4. Performance as a function of the number of iteration for different neural architectures in the pursuit domain with 200 agents. At least 16 agents need to occupy the same cell to capture an evader.

Table 3. Average returns on small-scale pursuit problem

	PS-TRPO	FIB
Average Returns	9.36 ± 0.52	9.29 ± 0.65

occupy the same location as the evader in order to catch it and obtain a reward. This problem has a total of 15625 states and 729 observations. The results comparing the average performance and their standard errors of PS-TRPO and FIB policies averaged over 100 simulations are shown in Table 3. The results demonstrate that PS-TRPO performs as well as the traditional approaches on the small problem, and has the ability to scale to large and continuous spaces.

6.2 Continuous Control Tasks

We next compared the performance of our algorithms on continuous control tasks. We compared the proposed training schemes with TRPO and found that parameter sharing and concurrent approaches tend to outperform centralized training for continuous tasks (Figs. 2b, c and d). GRU policies outperform MLP policies in the multi-walker and multi-ant domains. However, MLP policies perform significantly better in the waterworld domain. We believe this is caused by the difficulty of training recurrent networks compared to simpler feedforward ones with high-dimensional observations, especially when the task is relatively simple and does not require a history of observations. Visualizing the best performing policies showed consistent intelligent behavior in coordination between agents. In the waterworld domain, the pursuers learn to herd the evaders. In the multi-walker domain, the walkers learn to push the box forward without letting it fall down. In the multi-ant domain, the legs learn to avoid collision with each other.

Table 4. Average returns (over 50 runs) for policies trained with parameter sharing. DQN for discrete environment, DDPG for continuous

Task	PS-DQN/DDPG	PS-A3C	PS-TRPO
Pursuit	10.1 ± 6.3	25.5 ± 5.4	17.4 ± 4.9
Waterworld	NA	10.1 ± 5.7	49.1 ± 5.7
Multiwalker	-8.3 ± 3.2	12.4 ± 6.1	58.0 ± 4.2
Multi-ant	307.2 ± 13.8	483.4 ± 3.4	488.1 ± 1.3

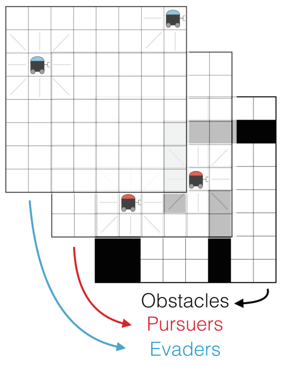


Fig. 5. Image like representation of an observation in the pursuit evasion domain. The locations of each entity (pursuers, evaders, and obstacles) are represented as bitmaps in their respective channels.

We also compared local and global reward schemes in the continuous domain (see Table 2). Overall, local reward shaping leads to better performance, and is critical to learning intelligent behavior in the waterworld and multi-walker domains (Table 2).

Finally, we compared the performance of PS-TRPO, PS-A3C, and PS-DDPG in continuous multi-agent domains. Training curves comparing PS-DDPG and PS-TRPO are shown in Fig. 3 for the multi-walker task, while the performance of all the algorithms and tasks are compared in Table 4. The results show the PS-TRPO significantly outperforms both PS-A3C and PS-DDPG in the waterworld and multi-walker domains. The performance of PS-TRPO and PS-A3C is comparable in the multi-ant domain.

6.3 Scaling

We next studied how well the parameter sharing method scales to larger observation spaces and many agents.

Curriculum training: Figure 6 shows the degrading performance of all policies with increasing number of agents in the multi-walker domain, and the performance improvements when curriculum learning is used. The policies were all

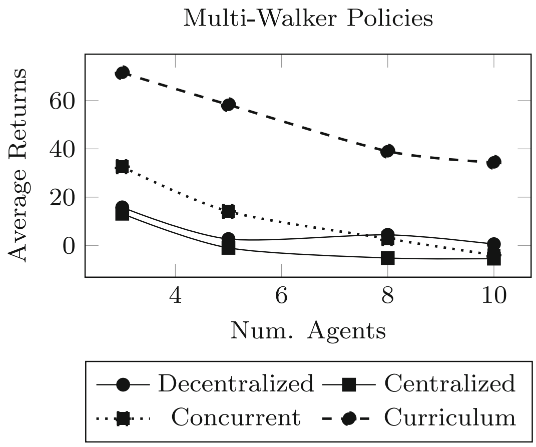


Fig. 6. Performance of multi-walker policies as a function of the number of agents during training. Each data point in the shared parameters, centralized, and concurrent curves was generated by training and evaluating a policy with a fixed number of agents. The curriculum curve was generated by evaluating a single policy with varying number of agents.

trained with TRPO. The decrease in performance is in part due to the increasing difficulty of the reinforcement learning task as the number of cooperating agents grows. As the number of agents required to complete a task increases, it becomes more difficult to reach the parts of the state space with positive rewards using naive exploration policies.

We investigated how a curriculum learning scheme can help scale the multi-walker problem in the number of agents. An intuitive curriculum for this problem is over the number of agents, and so we define a curriculum with the number of agents in the environment ranging from 2 to 10. Because the policies are decentralized even though the parameters are shared, they can be evaluated on tasks with any number of cooperating agents regardless of the number of cooperating agents present during training. Unfortunately, we found that these decentralized shared parameter policies trained on a few agents often fail to generalize to larger numbers of agents. We therefore define a Dirichlet distribution for this range of tasks with higher probability assigned to the simplest task (with 2 agents for Multi-Walker domain). We then sample an environment from this distribution over the tasks in the curriculum and optimize the policy with PS-TRPO for a few iterations. Once the expected reward for the most likely environment reaches a threshold, we change the distribution such that the next environment is most likely. We continue this curriculum until the expected reward in all environments reaches the defined threshold. Algorithm 2 describes this process. As shown earlier, the resulting policy outperforms policies trained without the curriculum. We believe this improvement in performance is due to two reasons: 1. The distribution over environments provides a regularization effect, helping avoid local

Algorithm 2. Curriculum Training

```
**Input:** Curriculum $\mathcal{T}$, Iteration $n$, Policy $\pi_\Theta$, $r_{\text{threshold}}$
$\alpha_\mathcal{T} \leftarrow [\text{length}(\mathcal{T}), 1, 1, \ldots]$
**while** $r_{\text{min}} < r_{\text{threshold}}$ **do**
    {Sample task from the task distribution.}
    $w \sim \text{Dirichlet}(\alpha_\mathcal{T})$
    $i \sim \text{Categorical}(w)$
    {Apply optimization step for a few iterations.}
    PS-TRPO $(\mathcal{T}_i, \pi_\theta, n)$
    {$e_{\text{curr}}$ is the task with the highest weight $\alpha_\mathcal{T}$.}
    $r_{e_{\text{curr}}} \leftarrow \text{Evaluate}(\pi_\theta, e_{\text{curr}})$
    **if** $r_{e_{\text{curr}}} > r_{\text{threshold}}$ **then**
        Circular shift $\alpha_\mathcal{T}$ weights to the next task
    {Find the minimum average reward across tasks.}
    $r_{\text{min}} \leftarrow \min_\mathcal{T} \mathbb{E} r_\mathcal{T}$
```

minima during optimization, and 2. It partially addresses the exploration problem by smoothly increasing the difficulty of the policy to be learned.

One potential issue with this experiment is that the curriculum scheme observed more episodes than the ones without curriculum. However, we tried training several policies with a fixed number of agents without a curriculum for an equivalent number of episodes. These policies converged before reaching the performance seen with curriculum training.

7 Conclusion

Despite the advances in decentralized control and reinforcement learning over recent years, learning cooperative policies in multi-agent systems remains a challenge. The difficulties lie in scalability to high-dimensional observation spaces and to large numbers of agents, accommodating partial observability, and handling continuous action spaces. In this work, we extended three deep reinforcement learning algorithms to the cooperative multi-agent context, and applied them to four high-dimensional, partially observable domains with many agents.

Our empirical evaluations show that PS-TRPO policies have substantially better performance than PS-DDPG and PS-A3C in continuous action collaborative multi-agent domains while PS-A3C is able to outperform PS-TRPO in the discrete domain. We suspect that DQN and DDPG perform poorly in systems with multiple learners due to the non-stationarity of the system dynamics caused by the changing policies of the agents. The non-stationary nature of the system makes experience replay samples obsolete and negatively impacts training. As evidence, we found that by disabling experience replay and instead relying on asynchronous training [14] we were able to improve on the performance of DQN and DDPG. However, we believe more hyperparameter tuning might be required to reduce the gap in overall performance in continuous domains with respect to TRPO. Finally, we presented how cooperative domains can form a natural curriculum over the number of agents required to collaborate on a task and discovered how this not only allows us to scale PS-TRPO to environments with large number of cooperating agents, but owing to the regularization effect offered, allows us to reach better local optima in general.

There are several areas for future work. To improve scalability of the proposed approach for larger numbers of cooperating agents further future work is needed. Two major challenges in multi-agent systems are accommodating reward sparsity through intelligent domain exploration and incorporating high-level task abstractions and hierarchy [39]. These are acute forms of similar challenges in the single agent learning. Recently, curiosity based information gain maximizing exploration strategy was explored by [40] . Similar ideas could be adapted to maximize information gain not only about the environment's dynamics, but the dynamics of an agent's behavior as well. Correspondingly, hierarchical value functions were integrated with deep reinforcement learning [41]. Incorporating task hierarchies in a multi-agent system would allow us to tackle learning specialization and heterogeneous behavior.

Acknowledgements. This work was supported by Army AHPCRC grant W911NF-07-2-0027. The authors would like to thank the anonymous reviewers for their helpful comments.

References

1. Tan, M.: Multi-agent reinforcement learning: independent vs. cooperative agents. In: International Conference on Machine Learning (ICML), pp. 330–337 (1993)
2. Panait, L., Luke, S.: Cooperative multi-agent learning: the state of the art. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS), vol. 11(3), pp. 387–434 (2005)
3. Bloembergen, D., Tuyls, K., Hennes, D., Kaisers, M.: Evolutionary dynamics of multi-agent learning: a survey. J. Artif. Intell. Res. **53**, 659–697 (2015)
4. Amato, C., Chowdhary, G., Geramifard, A., Ure, N.K., Kochenderfer, M.J.: Decentralized control of partially observable Markov decision processes. In: IEEE Conference on Decision and Control (CDC), Florence, Italy (2013)
5. Bernstein, D.S., Zilberstein, S., Immerman, N.: The complexity of decentralized control of Markov decision processes. In: Conference on Uncertainty in Artificial Intelligence (UAI), pp. 32–37 (2000)
6. Banerjee, B., Lyle, J., Kraemer, L., Yellamraju, R.: Sample bounded distributed reinforcement learning for decentralized POMDPs. In: AAAI Conference on Artificial Intelligence (AAAI) (2012)
7. Omidshafiei, S., Agha-mohammadi, A.-A., Amato, C., Liu, S.-Y., How, J.P., Vian, J.: Graph-based cross entropy method for solving multi-robot decentralized POMDPs. In: IEEE International Conference on Robotics and Automation (ICRA) (2016)
8. Tesauro, G.: Extending Q-learning to general adaptive multi-agent systems. In: Advances in Neural Information Processing Systems (NIPS) (2003)
9. Lin, L.-J.: Reinforcement learning for robots using neural networks, Ph.D. dissertation. Carnegie Mellon University (1992)

10. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
11. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. J. Mach. Learn. **17**(39), 1–40 (2016)
12. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International Conference on Machine Learning (ICML) (2015)
13. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971 (2015)
14. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning, arXiv preprint arXiv:1602.01783 (2016)
15. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: International Conference on Machine Learning (ICML), pp. 41–48 (2009)
16. Busoniu, L., Babuska, R., Schutter, B.D.: Multi-agent reinforcement learning: a survey. In: International Conference on Control, Automation, Robotics and Vision, vol. 527, pp. 1–6 (2006)
17. Ono, N., Fukumoto, K.: A modular approach to multi-agent reinforcement learning. In: Weiß, G. (ed.) LDAIS/LIOME -1996. LNCS, vol. 1221, pp. 25–39. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-62934-3_39
18. Guestrin, C., Lagoudakis, M., Parr, R.: Coordinated reinforcement learning. In: International Conference on Machine Learning (ICML), vol. 2, pp. 227–234 (2002)
19. Lauer, M., Riedmiller, M.: An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In: International Conference on Machine Learning (ICML), pp. 535–542 (2000)
20. Singh, S.P., Jaakkola, T.S., Jordan, M.I.: Learning without state-estimation in partially observable markovian decision processes. In: International Conference on Machine Learning (ICML) (1994)
21. Peshkin, L., Kim, K.-E., Meuleau, N., Kaelbling, L.P.: Learning to cooperate via policy search. In: Conference on Uncertainty in Artificial Intelligence (UAI), pp. 489–496 (2000)
22. Fernández, F., Parker, L.E.: Learning in large cooperative multi-robot domains. Int. J. Robot. Autom. **16**(4), 217–226 (2001)
23. Tamakoshi, H., Ishii, S.: Multiagent reinforcement learning applied to a chase problem in a continuous world. Artif. Life Robot. **5**(4), 202–206 (2001)
24. Das, A.K., Fierro, R., Kumar, V., Ostrowski, J.P., Spletzer, J., Taylor, C.J.: A vision-based formation control framework. IEEE Trans. Robot. Autom. **18**(5), 813–825 (2002)
25. Cortes, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. In: IEEE International Conference on Robotics and Automation (ICRA), vol. 2, pp. 1327–1332. IEEE (2002)
26. Olfati-Saber, R., Fax, J.A., Murray, R.M.: Consensus and cooperation in networked multi-agent systems. Proc. IEEE **95**(1), 215–233 (2007)
27. Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., Vicente, R.: Multiagent cooperation and competition with deep reinforcement learning, arXiv preprint arXiv:1511.08779 (2015)
28. Foerster, J.N., Assael, Y.M., de Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. In: Advances in Neural Information Processing Systems (NIPS) (2016)

29. Sukhbaatar, S., Szlam, A., Fergus, R.: Learning multiagent communication with backpropagation. In: Advances in Neural Information Processing Systems (NIPS) (2016)
30. Ng, A.Y., Harada, D., Russell, S.: Policy invariance under reward transformations: theory and application to reward shaping. In: International Conference on Machine Learning (ICML), vol. 99, pp. 278–287 (1999)
31. Bagnell, D., Ng, A.Y.: On local rewards and scaling distributed reinforcement learning. In: Advances in Neural Information Processing Systems, pp. 91–98 (2005)
32. Vidal, R., Shakernia, O., Kim, H.J., Shim, D.H., Sastry, S.: Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. IEEE Trans. Robot. Autom. **18**(5), 662–669 (2002)
33. Ho, J., Gupta, J.K., Ermon, S.: Model-free imitation learning with policy optimization. In: International Conference on Machine Learning (ICML) (2016)
34. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym (2016)
35. Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438 (2015)
36. Tieleman, T., Hinton, G.: Lecture 6.5-RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Netw. Mach. Learn. **4**, 26–31 (2012)
37. Nair, R., Tambe, M., Yokoo, M., Pynadath, D., Marsella, S.: Taming decentralized POMDPs: towards efficient policy computation for multiagent settings. In: International Joint Conference on Artificial Intelligence (IJCAI) (2003)
38. Hauskrecht, M.: Incremental methods for computing bounds in partially observable Markov decision processes. In: AAAI Conference on Artificial Intelligence (AAAI) (1997)
39. Parr, R., Russell, S.: Reinforcement learning with hierarchies of machines. In: Advances in Neural Information Processing Systems (NIPS), pp. 1043–1049 (1998)
40. Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., Abbeel, P.: Variational information maximizing exploration. arXiv preprint arXiv:1605.09674 (2016)
41. Kulkarni, T.D., Narasimhan, K.R., Saeedi, A., Tenenbaum, J.B.: Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation. arXiv preprint arXiv:1604.06057 (2016)