A Decentralized Policy Gradient Approach to Multi-Task Reinforcement Learning

Sihan Zeng[1] **Malik Aqeel Anwar**[1] **Thinh T. Doan**[2] **Arijit Raychowdhury**[1] **Justin Romberg**[1]

[1]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA
[2]Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg Virginia, USA

Abstract

We develop a mathematical framework for solving multi-task reinforcement learning (MTRL) problems based on a type of policy gradient method. The goal in MTRL is to learn a common policy that operates effectively in different environments; these environments have similar (or overlapping) state spaces, but have different rewards and dynamics. We highlight two fundamental challenges in MTRL that are not present in its single task counterpart, and illustrate them with simple examples. We then develop a decentralized entropy-regularized policy gradient method for solving the MTRL problem, and study its finite-time convergence rate. We demonstrate the effectiveness of the proposed method using a series of numerical experiments. These experiments range from small-scale "GridWorld" problems that readily demonstrate the trade-offs involved in multi-task learning to large-scale problems, where common policies are learned to navigate an airborne drone in multiple (simulated) environments.

1 INTRODUCTION

In reinforcement learning (RL), an agent tries to learn an optimal policy through repeated interactions with its environment, modeled as a Markov decision process (MDP), with the goal of optimizing its long-term cumulative rewards. Combined with powerful function approximation such as neural networks, (deep) reinforcement learning has received great successes in solving challenging problems in different applications, including game playing [Mnih et al., 2015, Silver et al., 2016, OpenAI et al., 2019], healthcare [Yu et al., 2019, Esteva et al., 2019], robotics [Kober et al., 2013, Haarnoja et al., 2019], and autonomous navigation [Kretzschmar et al., 2016, Zhu et al., 2017, Anwar and Raychowdhury, 2020]. These results, however, are primarily achieved only on a single task, and every new task almost requires the agent to be re-trained from scratch.

Multi-task reinforcement learning (MTRL) addresses this problem by finding a single policy that is simultaneously effective for a number of tasks. We are interested in developing a new method for MTRL by using a group of learning agents. We consider a scenario where multiple agents, each learning in its own environment, work together to learn a common policy by sharing their policy parameters. This common policy may perform slightly worse than the optimal policy for each local task, but is general enough to solve all local tasks reasonably well. In other words, the agent learns how to perform well not only in its own environment but also in unseen environments explored by other agents.

Existing approaches to solving problems of this nature [Hessel et al., 2019, Espeholt et al., 2018, Yu et al., 2020] typically use a specific "master/worker" model for agent interaction, where worker agents independently collect observations in their respective environments, which are then summarized (perhaps through a gradient computation) and reported to a central master. We are interested in understanding MTRL under a more flexible, decentralized communication model where agents only share information with a small subset of other agents. This framework is inspired by applications where centralized coordination is unwieldy or impossible; one example would be a network of mobile robots exploring different parts of an area of interest that can only communicate locally. This question has not yet been addressed in the existing literature, and our focus, therefore, is to solve this important problem by developing a decentralized policy gradient method.

Main Contributions

• We present a clean mathematical formulation for MTRL problems over a network of agents, where each task is assigned to a single agent. Framing the problem in the language of distributed optimization allows us to develop a decentralized policy gradient algorithm that finds a single

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

policy that is effective for each of the tasks.

• We present, in Section 2.2, two simple examples that illustrate the fundamental differences between learning a policy for one task and learning for multiple tasks.

• We provide theoretical guarantees for the performance of our decentralized policy gradient algorithm. We show that in the tabular setting, the algorithm converges to a stationary point of the global (non-concave) objective. Under a further assumption on the structure of environments' dynamics, the algorithm converges to the globally optimal value.

• We demonstrate the effectiveness of the proposed method using numerical experiments on challenging MTRL problems. Our small-scale "Grid World" problems, which can be reliably solved using a complete tabular representation for the policy, demonstrate how the decentralized policy gradient algorithm balances the interests of the agents in different environments. Our experiments for learning to navigate airborne drones in multiple (simulated) environments show that the algorithm can be scaled to problems that require significant amounts of data and use neural network representations for the policy function.

1.1 RELATED WORKS

In recent years, multi-task RL has become an emerging topic as a way to scale up RL solutions. This topic has received a surge of interests, and a number of solutions have been proposed for solving this problem, including policy distillation [Rusu et al., 2015, Traoré et al., 2019], distributed RL algorithms over actors/learner networks [Espeholt et al., 2018, Hessel et al., 2019, Liu et al., 2016, Yu et al., 2020], and transfer learning [Gupta et al., 2017, D'Eramo et al., 2020]. Distributed parallel computing has also been applied to speed up RL algorithms for solving single task problems [Mnih et al., 2016, Nair et al., 2015, Assran et al., 2019].

Similar to our work, Espeholt et al. [2018], Hessel et al. [2019] also aim to solve MTRL with policy gradient algorithms in a distributed manner. These works propose sharing the local trajectories/data collected by workers in each environment to a centralized server where learning takes place. When the data dimension is large, the amount of information required to be exchanged could be enormous. In contrast, exchanging the policy parameters could be a more compact and efficient form of communication in applications with a large state representation but a much smaller policy representation. Moreover, we observe that a wide range of practical problems do not allow for a centralized communication topology [Ovchinnikov et al., 2014]. Motivated by these observations, we consider a decentralized policy gradient method where the agents only exchange their policy parameters according to a decentralized communication graph. This makes our work fundamentally different from the existing literature. Indeed, our work can be considered as a decentralized and multi-task variant of the policy gradient method studied in Agarwal et al. [2020], where the authors consider a single-task RL.

Other works in meta-learning and transfer learning also essentially aim to achieve MTRL, where these two methods essentially attempt to reduce the resources required to learn a new task by utilizing related existing information; see for example Wang et al. [2016], Nagabandi et al. [2018], Anwar and Raychowdhury [2020]. Our work is fundamentally different from these papers, where we address MTRL by leveraging the collaboration between a number of agents.

We also note some relevant works on decentralized algorithms in multi-agent reinforcement learning (MARL), where a group of agents operate in a common environment and aim to solve a single task [Zhang et al., 2018, Chu et al., 2020, Qu and A. Wierman, 2019, Doan et al., 2019, Ding et al., 2019, Li et al., 2020, Wai et al., 2018, Kar et al., 2013, Lee et al., 2019, Zhang et al., 2019]. The setting in these work is different from ours since we consider multi-task RL, which is more challenging than solving a single task.

2 MULTI-TASK REINFORCEMENT LEARNING

We consider an MTRL problem with N agents operating in N different environments. These environments, each characterized by a different Markov decision process (MDP) as described below, might also be interpreted as encoding a different task that an agent attempts to accomplish. Although each agent acts and makes observations in a single environment, their goal is to learn a policy that is jointly optimal across all of the environments. Information is shared between the agents through connections described by edges in an undirected graph. We do not require the state spaces to be the same in each of the environments; in general, the learned joint policy is a mapping from the union of state spaces to the action space.

2.1 MTRL FORMULATION

The MDP at agent i is given by the 5-tuple $\mathcal{M}_i = (\mathcal{S}_i, \mathcal{A}, \mathcal{P}_i, \mathcal{R}_i, \gamma_i)$ where $\mathcal{S}_i$ is the set of states, $\mathcal{A}$ is the set of possible actions, which has to be common across tasks, $\mathcal{P}_i$ is the transition probabilities that specify the distribution on the next state given the current state and an action, $\mathcal{R}_i : \mathcal{S}_i \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma_i \in (0, 1)$ is the discount factor. We denote by $\mathcal{S} = \cup_i \mathcal{S}_i$, where $\mathcal{S}_i$ can share common states. We focus on randomized stationary policies (RSPs), where agent i maintains a policy π_i that assigns to each $s \in \mathcal{S}_i$ a probability distribution $\pi_i(\cdot|s)$ over $\mathcal{A}$.

Given a policy π, let V_i^π be the value function associated

with the i-th environment,

$$V_i^\pi(s_i) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma_i^k \mathcal{R}_i(s_i^k, a_i^k) \,|\, s_i^0 = s_i\right], \; a_i^k \sim \pi(\cdot|s_i^k). \tag{1}$$

Similarly, we denote by Q_i^π and A_i^π the Q-function and advantage function in the i-th environment

$$Q_i^\pi(s_i, a_i) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma_i^k \mathcal{R}(s_i^k, a_i^k) \,|\, s_i^0 = s_i, a_i^0 = a_i\right],$$
$$A_i^\pi(s_i, a_i) = Q_i^\pi(s_i, a_i) - V_i^\pi(s_i). \tag{2}$$

Without loss of generality, we assume that $\mathcal{R}_i(s, a) \in [0, 1]$, implying for any policy π and $\forall s \in \mathcal{S}_i, a \in \mathcal{A}$

$$0 \le V_i^\pi(s) \le \frac{1}{1-\gamma_i}, \quad -\frac{1}{1-\gamma_i} \le A_i^\pi(s, a) \le \frac{1}{1-\gamma_i}. \tag{3}$$

Let ρ_i be an initial state distribution over $\mathcal{S}_i$, and with some abuse of notation we denote the long-term reward associated with this distribution as $V_i^\pi(\rho_i) = \mathbb{E}_{s_i \sim \rho_i}[V_i^\pi(s_i)]$.

To parameterize the policy, we consider the scenario where each agent maintains $\theta_i \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ and uses the popular softmax parameterization[1], i.e.

$$\pi_{\theta_i}(a \,|\, s) = \frac{\exp(\theta_{i\,;\,s,a})}{\sum_{a' \in \mathcal{A}} \exp(\theta_{i\,;\,s,a'})}. \tag{4}$$

The goal of the agents is to cooperatively find a parameter θ^* that maximizes the total cumulative discounted rewards

$$\theta^* = \arg\max_\theta V^{\pi_\theta}(\boldsymbol{\rho}) \triangleq \sum_{i=1}^{N} V_i^{\pi_\theta}(\rho_i), \;\; \boldsymbol{\rho} = [\rho_1; \ldots; \rho_N], \tag{5}$$

which is a non-concave objective [Agarwal et al., 2020].

Treating each of the environments as independent RL problems would produce different policies π_i^*, each maximizing their respective V_i^π. Our focus in this paper is to find a single π^* that balances the performance across all environments.

2.2 MTRL CHALLENGES

While solving single task RL is well understood at least in the tabular settings, MTRL is more challenging than it appears from (5). Here, we provide two fundamental challenges of MTRL, which make this problem much more difficult than its single-task counterpart.

Deterministic vs stochastic policies. In single task RL it is known that under mild assumptions there exists a deterministic policy π^* that maximizes the objective [Puterman, 1994]. In addition, the value function of the optimal deterministic policy satisfies the Bellman optimality equation, motivating the development of the popular Q-learning method. In

[1]Note that our method also works with other forms of stochastic policies.

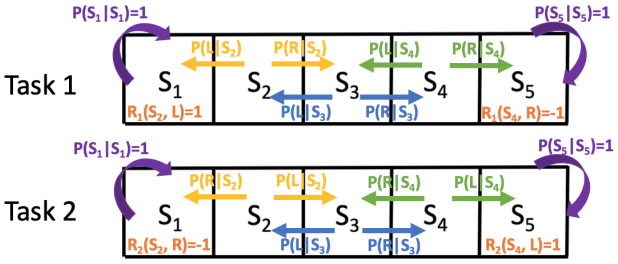


Figure 1: 2-Task GridWorld Problem

MTRL, where each task operates under different dynamics (transition probability matrices), there need not be an optimal deterministic policy, and hence there is no natural analog to the Bellman equation. We illustrate this below with a simple "GridWorld" example.

In the two-task GridWorld problem shown in Fig. 1, there are two environments with the same state and action spaces. The dynamics and reward functions, however, are different. The two actions, labeled L and R, deterministically move the agents to the left and right, respectively, in all states in Task 1. In Task 2, the effect of L and R is reversed for states S_2 and S_4: applying L (R) in S_2 transitions to S_3 (S_1), while applying L (R) in S_4 transitions to S_5 (S_3). In both environments, the agents stay in states S_1 and S_5 when they reach them. In Task 1 there is a reward of $+1$ for reaching S_1 and a penalty of -1 for reaching S_5; these rewards are reversed for Task 2.

We now consider what happens when we are asked to find a single policy that maximizes the sum of the cumulative rewards of the two tasks. It is obvious that the optimal policy for state S_2 and S_4 is to always take action L in order to reach the positive reward or to stay away from the negative reward. The only state whose optimal policy remains unclear is S_3. With the detailed computation left to the supplementary material, we find that the optimal (stochastic) policy π^* is

$$\pi^*(a|S_3) = \begin{cases} 0.5, & a = L, \\ 0.5, & a = R, \end{cases}$$

which yields $V^{\pi^*}(S_3) = \frac{2\gamma}{2-\gamma^2}$.

By symmetry, the two possible deterministic policies

$$\pi_l(a|S_3) = \begin{cases} 1, & a = L \\ 0, & a = R \end{cases} \text{ and } \pi_r(a|S_3) = \begin{cases} 0, & a = L \\ 1, & a = R \end{cases}$$

produce the same value for state S_3. For example, if the agent takes π_l as an action in S_3, it keeps moving left in task 1 and oscillates between S_2 and S_3 in task 2. In this case, one can show that due to the discount factor $V^{\pi_l}(S_3) = \gamma$. A similar argument holds for the case when the agent takes π_r in S_3, where we have $V^{\pi_r}(S_3) = \gamma$. In both cases, $V^{\pi_l}(S_3) = V^{\pi_r}(S_3) < V^{\pi^*}(S_3)$ when $\gamma > 0$, which implies that any deterministic policy is sub-optimal.

As a consequence, RL methods that implicitly rely on the existence of a deterministic optimal policy (e.g., Q learning)

cannot solve this type of problem. This provides additional motivation for us to study randomized policies and take on a policy gradient approach.

Gradient domination condition. In single task RL, it has been shown that the objective function, despite being non-concave, satisfies a kind of "gradient domination" condition [Agarwal et al., 2020], which implies that every stationary point is globally optimal. This is important as it guarantees that the policy gradient algorithm, by reaching a stationary point in single task RL, can find the globally optimal policy. As we show below, in the multi-task problem we cannot expect to have this condition in the general setting. The landscape of the MTRL objective is so irregular that there could exist multiple stationary points which are not global optima. We illustrate this issue with another simple example below.

Let us consider again the 2-task GridWorld problem in Fig.1. Here we make a slight modification to the dynamics of the tasks. In task 1 and task 2, regardless of the action taken in state S_2 and S_4, the transition probability is

$$P_1(s|S_2) = \begin{cases} 1-p, & s = S_1 \\ p, & s = S_3 \end{cases}$$
$$P_1(s|S_4) = \begin{cases} 1-p, & s = S_3 \\ p, & s = S_5 \end{cases}$$
$$P_2(s|S_2) = \begin{cases} p, & s = S_1 \\ 1-p, & s = S_3 \end{cases}$$
$$P_2(s|S_4) = \begin{cases} p, & s = S_3 \\ 1-p, & s = S_5 \end{cases}$$

for some $0.5 < p \leq 1$.

It is obvious that the policy gradient for states S_2 and S_4 will always be zero as the value function is a constant in terms of the policy at these two states. We only have to optimize the policy for state S_3.

Under the softmax parameterization, we maintain parameters $\theta_{S_3,L}$ and $\theta_{S_3,R}$ to encode the policy

$$\pi_\theta(L|S_3) = \frac{e^{\theta_{S_3,L}}}{e^{\theta_{S_3,L}} + e^{\theta_{S_3,R}}} \text{ and } \pi_\theta(R|S_3) = \frac{e^{\theta_{S_3,R}}}{e^{\theta_{S_3,L}} + e^{\theta_{S_3,R}}}.$$

We consider the case where the agents always start from state S_3. It can be shown that $\theta_{S_3,L} = 1, \theta_{S_3,R} = \infty$ (always taking action R) and $\theta_{S_3,L} = \infty, \theta_{S_3,R} = 1$ (always taking action L) are both stationary points and achieve the global maximum of the objective (5), while $\theta_{S_3,L} = 1, \theta_{S_3,R} = 1$ (taking action L and R each with probability 0.5) is a non-globally optimal stationary point[2]. When gradient based methods are used to optimize (5), it could be trapped at the stationary points without finding the global optimality. In Section 4.1 and 4.2, we dive deeper into the problem and show that globality optimality can be reached under a restrictive structural assumption.

3 DECENTRALIZED POLICY GRADIENT

One advantage of using softmax policies is that θ is unconstrained, making (5) an unconstrained optimization problem. One may attempt to apply (stochastic) gradient ascent and utilize the existing standard techniques in (stochastic) optimization to analyze its performance. However, the optimal policy, which is possibly deterministic, may be attained only by sending θ to infinity. Such an exponential scaling with the parameters makes studying the convergence of this method more challenging. To handle this challenge, a common approach in the literature is to utilize the entropy-based regularization [Mnih et al., 2016, Agarwal et al., 2020]. In this paper, we use the relative-entropy as a regularization for the objective in (5) inspired by Agarwal et al. [2020]. Specifically, the relative-entropy of π_θ is given as

$$\begin{aligned} \text{RE}(\pi_\theta) &\triangleq \mathbb{E}_{s\sim\text{Unif}_\mathcal{S}}\left[\text{KL}\left(U_\mathcal{A}, \pi_\theta(\cdot|s)\right)\right] \\ &= -\frac{1}{|\mathcal{S}||\mathcal{A}|}\sum_{a\in\mathcal{A}} \log \pi_\theta(a \mid s) - \log|\mathcal{A}|, \end{aligned} \tag{6}$$

where $U_\mathcal{A}$ is the uniform distribution over $\mathcal{A}$ and $\text{KL}(p,q) = \mathbb{E}_{x\sim p}[-\log(q(x)/p(x))]$. The relative-entropy regularized variant of (5) is then given as

$$L^\lambda(\theta;\boldsymbol{\rho}) = \sum_{i=1}^N L_i^\lambda(\theta;\rho_i) = \sum_{i=1}^N \left(V_i^{\pi_\theta}(\rho_i) - \lambda \text{RE}\left(\pi_\theta\right)\right), \tag{7}$$

where λ is a regularization parameter. Defining the discounted state visitation distribution $d_i^{\pi_\theta}$ under a policy π_θ in the i-th environment

$$d_i^{\pi_\theta}(s \mid s_0) \triangleq (1-\gamma_i)\sum_{k=0}^\infty \gamma_i^k P_i^{\pi_\theta}(s_i^k = s \mid s_i^0 = s_0),$$

and $d_{i,\rho_i}^{\pi_\theta}(s) = \mathbb{E}_{s_0\sim\rho_i}[d_i^{\pi_\theta}(s \mid s_0)]$. The gradient of L_i^λ is [3]

$$\begin{aligned} \frac{\partial L_i^\lambda(\theta;\rho_i)}{\partial \theta_{s,a}} &= \frac{1}{1-\gamma_i} d_{i,\rho_i}^{\pi_\theta}(s)\pi_\theta(a \mid s) A_i^{\pi_\theta}(s,a) \\ &\quad + \frac{\lambda}{|\mathcal{S}|}\left(\frac{1}{|\mathcal{A}|} - \pi_\theta(a|s)\right). \end{aligned} \tag{8}$$

Our focus now is to apply gradient ascent methods for optimizing L^λ in a decentralized setting. In fact, under some proper choice of λ the proposed algorithm can get arbitrarily close to the stationary point of problem (5).

To optimize (7), we use the decentralized policy gradient method formally stated in Algorithm 1. In this algorithm,

[2] Derivation details can be found in Section A of the supplementary material.

[3] The derivation is in Section B.4 of the supplementary material.

each agent can communicate with each other through an undirected and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where agents i and j can exchange messages if and only if they are connected in $\mathcal{G}$. We denote by $\mathcal{N}_i = \{j : (i,j) \in \mathcal{E}\}$ the set of agent i's neighbors. In addition, each agent i maintains its own local policy parameter θ_i, as an estimate of the optimal θ^* of (5). Finally, μ_i is the local initial distribution at agent i, which can be chosen differently from ρ_i.

Algorithm 1: Decentralized Policy Gradient Algorithm

Initialization: Each agent i initializes $\theta_i^0 \in \mathbb{R}^d$, an initial distribution μ_i, and step sizes $\{\alpha^k\}_{k\in\mathbb{N}}$.;

for *k=1,2,3,...* **do**

Each agent i simultaneously implements:

1) Exchange θ_i^k with neighbors $j \in \mathcal{N}_i$

2) Compute the gradient g_i^k of $L_i^\lambda(\theta_i^k; \mu_i)$

3) Policy update:

$$\theta_i^{k+1} = \sum_{j\in\mathcal{N}_i} W_{ij}\theta_j^k + \alpha^k g_i^k. \tag{9}$$

end

At any time $k \geq 0$, agent i first exchanges its iterates with its neighbors $j \in \mathcal{N}_i$ and compute the gradient g_i^k of $L_i^\lambda(\theta_i^k; \mu_i)$ only using information from its environment. Agent i updates θ_i by implementing (9), where it takes a weighted average of θ_i^k with θ_j^k received from its neighbors $j \in \mathcal{N}_i$, following by a local gradient step. The goal of this weighted average is to achieve a consensus among the agents' parameters, i.e., $\theta_i = \theta_j$, while the local gradient steps are to push this consensus point toward the optimal θ^*. Here, W_{ij} is some non-negative weight, which i assigns for θ_j^k. The conditions on W_{ij} to guarantee the convergence of Algorithm 1 are given in the next section.

4 CONVERGENCE ANALYSIS

In this section, our focus is to study the performance of Algorithm 1 under the tabular setting, i.e., $\theta \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$. It is worth recalling that each function V_i^π in (5) is in general non-concave. To show the convergence of our algorithm, we study the case when g_i is an exact estimate of ∇L_i^λ (Eq. (8)), and consider the weight matrix W satisfying the following assumption, which is fairly standard in the literature of decentralized consensus-based optimization [Zhang et al., 2018, Doan et al., 2019, Zeng et al., 2020].

Assumption 1. *Let $W = [W_{ij}] \in \mathbb{R}^{N\times N}$ be a doubly stochastic matrix, i.e., $\sum_i W_{ij} = \sum_j W_{ij} = 1$, with $W_{ii} > 0$. Moreover, $W_{ij} > 0$ iff i and j are connected, otherwise $W_{ij} = 0$.*

We denote by σ_2 and σ_N the second largest and the smallest singular values of W, respectively. Our first main result shows that the algorithm converges to the stationary point of (5) at a rate $\mathcal{O}(1/\sqrt{K})$, where $\mu_i = \rho_i$, for all i.

Theorem 1. *Suppose that Assumption 1 holds. Let $\{\theta_i^k\}$, for all i, be generated by Algorithm 1. In addition, let $\mu_i = \rho_i$, for all i, and the step sizes $\alpha^k = \alpha$ satisfying*

$$\alpha \in \Big(0, \frac{1+\sigma_N}{\sum_{i=1}^N \frac{16}{(1-\gamma_i)^3} + \frac{4N\lambda}{|\mathcal{S}|}}\Big). \tag{10}$$

Then $\forall i$, θ_i^k satisfies

$$\begin{aligned}&\min_{k<K} \Big\| \frac{1}{N}\sum_{j=1}^N \nabla V_j(\theta_i^k; \rho_j)\Big\|^2 \\ &\leq \mathcal{O}\Big(\frac{1}{K\alpha} + \frac{\alpha^2}{N(1-\sigma_2)^2\sum_{j=1}^N(1-\gamma_j)^6} + \frac{\lambda^2}{N}\Big).\end{aligned} \tag{11}$$

For an ease of exposition, we delay the analysis of this theorem to the supplementary material, where we provide an exact formula for the right-hand side of (11). First, our upper bound in (11) depends quadratically on the inverse of the spectral gap $1-\sigma_2$, which shows the impact of the graph $\mathcal{G}$ on the convergence of the algorithm. Second, this bound states that under a constant step size the norm of the gradient converges to a ball with radius $\mathcal{O}(\alpha)$ at a rate $\mathcal{O}(1/\sqrt{K})$. As the step size is reduced, we get closer to a stationary point of (5). This rate matches the one for single task RL in Agarwal et al. [2020]. However, while we only show the convergence to a stationary point, a global optimality is achieved there. Below, we provide insight on this discrepancy, which has already been suggested by the example in Section 2.2.

Remark 1. *we note that Algorithm 1 can be applied when the policy is represented by function approximations (e.g., neural networks). Under an function approximation, we can guarantee the convergence to a stationary point with the rate stated in Theorem 1 under an assumption on the Lipschitzness of the function approximation, and the proof of Theorem 1 in Section B.1 of the supplementary material still goes through. For conciseness, we do not formally state the theorem under function approximations, but will provide numerical simulations using neural networks in Section 5.*

4.1 TASK CONFLICTING

For single task RL, policy gradient methods can return a globally optimal policy in the tabular setting [Agarwal et al., 2020]. A natural question to ask is whether multi-task versions of the policy gradient methods can return a globally optimal policy in MTRL. Unfortunately, this question is no in general, which has been shown by the example in Section 2.2. Here we provide a more mathematical explanation of the issue. First, we consider single task RL with the softmax parameterization and relative entropy regularizer. In

this setting, by using policy gradient methods one can show that $\|\nabla_\theta L_i^\lambda(\theta;\mu_i)\| \to 0$; [Agarwal et al., 2020, Corollary 5.1], which translates to $A_i^{\pi_\theta}(s,a)$ decaying to zero for all s,a for a proper choice of λ. The decay of the advantage function is used to show the globally optimal convergence of policy gradient methods, where the objective function satisfies [Agarwal et al., 2020, Theorem 5.2]

$$V_i^{\pi^*}(\rho_i) - V_i^{\pi_\theta}(\rho_i) = \frac{1}{1-\gamma}\sum_{s,a} d_{i,\rho_i}^{\pi^*}(s)\pi^*(a|s)A_i^{\pi_\theta}(s,a).$$

This condition is often referred to as the gradient domination property, an analog of the Polyak-Lojasiewicz condition commonly used to show the global optimality in nonconvex optimatization. Unfortunately, we do not have this property in the general MTRL setting. Although each component task function $V_i^\pi(\rho_i)$ satisfies the gradient domination condition, their sum may not due to what we call distribution mismatch or task conflicting. To further explain this, using a step of analysis similar to Agarwal et al. [2020, Theorem 5.2], we have

$$\begin{aligned} &V(\theta^*;\boldsymbol{\rho}) - V(\theta;\boldsymbol{\rho}) \\ &= \sum_{i=1}^N \frac{1}{1-\gamma_i}\sum_{s\in\mathcal{S}_i}\sum_{a\in\mathcal{A}} d_{i,\rho_i}^{\pi_{\theta^\star}}(s)\pi_{\theta^\star}(a\,|\,s)A_i^{\pi_\theta}(s,a) \\ &= \sum_{s,a}\pi_{\theta^\star}(a\,|\,s)\sum_{i:s\in\mathcal{S}_i}\frac{d_{i,\rho_i}^{\pi_{\theta^\star}}(s)}{d_{i,\mu_i}^{\pi_\theta}(s)}\frac{d_{i,\mu_i}^{\pi_\theta}(s)}{1-\gamma_i}A_i^{\pi_\theta}(s,a). \end{aligned}$$

To achieve the global optimality one needs to have

$$\sum_{i:s\in\mathcal{S}_i}\frac{d_{i,\rho_i}^{\pi_{\theta^\star}}(s)}{d_{i,\mu_i}^{\pi_\theta}(s)}\frac{d_{i,\mu_i}^{\pi_\theta}(s)}{1-\gamma_i}A_i^{\pi_\theta}(s,a) \to 0,$$

while the policy gradient algorithm can only return (cf. (8))

$$\sum_{i:s\in\mathcal{S}_i}\frac{d_{i,\mu_i}^{\pi_\theta}(s)}{1-\gamma_i}A_i^{\pi_\theta}(s,a) \to 0.$$

With a similar line of analysis, one can show that this issue also arises in other forms of policy, e.g., direct parameterization, and with other types of policy gradient methods, e.g., mirror descent [Lan, 2021]. This problem is due to the ratios $d_{i,\rho_i}^{\pi_{\theta^\star}}(s)/d_{i,\mu_i}^{\pi_\theta}(s)$ being different across the environments. We call this ratio the distribution mismatch between the environments, representing the conflict between tasks. Because of this distribution mismatch, gradient descent approach uses biased gradients in its update. One cannot easily correct this mismatch since the optimal policy π_{θ^*} is unknown.

4.2 ACHIEVING GLOBAL OPTIMALITY

Despite the difficulty of the MTRL problem, we provide a sufficient condition on the structure of the MDPs, where a global optimality can be achieved by Algorithm 1.

Assumption 2. *Let π_{θ^*} be an optimal policy solving* (5). *Then for any π_θ and $\boldsymbol{\mu}$ we have*

$$\frac{d_{i,\rho_i}^{\pi_{\theta^*}}(s)}{d_{i,\mu_i}^{\pi_\theta}(s)} = \frac{d_{j,\rho_j}^{\pi_{\theta^*}}(s)}{d_{j,\mu_j}^{\pi_\theta}(s)},\ \forall s : s\in\mathcal{S}_i\cap\mathcal{S}_j,\ \forall i,j\in[N]. \quad (12)$$

We know that $d_{i,\rho_i}^{\pi_\theta}(s_i)$ (similarly, $d_{i,\mu_i}^{\pi_\theta}(s_i)$) is the discounted fraction of time that agent i visits state $s_i\in\mathcal{S}_i$ when using ρ_i (similarly, μ_i) as the initial distribution. Qualitatively, this assumption can be interpreted as enforcing that the joint states between the environments are equally explored. Mathematically, this assumption guarantees the objective function (5) obeys a kind of gradient domination when each function $V_i^\pi(\rho_i)$ satisfies this condition.

We note that Assumption 2 holds in the important case where the component tasks share the same state space and transition probability, but differ in their reward functions.

Under Assumption 2, we show that Algorithm 1 finds the global optimality of (5). For simplicity, we assume without loss of generality that $\theta_i^0 = \theta_j^0, \forall\, i,j$. Let $\alpha^k = \alpha$ satisfying

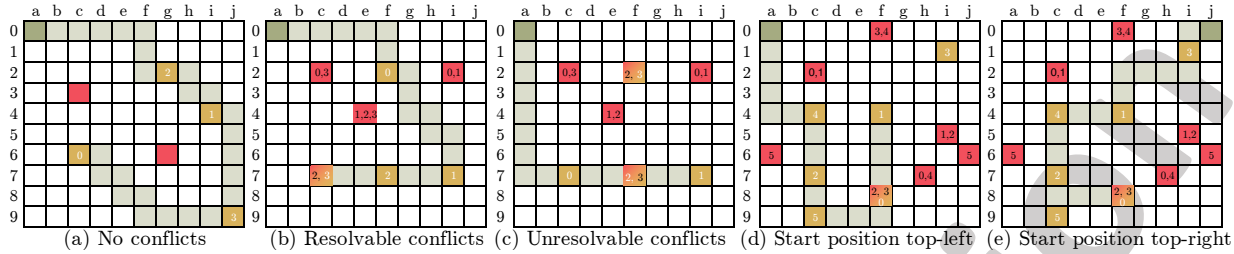
$$\alpha < \frac{1}{\sum_{i=1}^N\left(\frac{8}{(1-\gamma_i)^3}+\frac{2\lambda}{|\mathcal{S}|}\right)} \times \min\left\{1+\sigma_N\ ;\ \frac{\lambda N(1-\sigma_2)}{4|\mathcal{S}||\mathcal{A}|\left(2N\lambda+\sum_{i=1}^N\frac{1}{(1-\gamma_i)^2}\right)}\right\}. \quad (13)$$

Theorem 2. *Suppose that Assumptions 1 and 2 hold. Given an $\epsilon>0$, let $\lambda = \epsilon\,/\,2N\|d_{\boldsymbol{\rho}}^{\pi_{\theta^*}}/\boldsymbol{\mu}\|_\infty$ and α^k satisfy* (13). *Let θ^* be a solution of* (5). *Then $\forall i$, θ_i^k returned by Algorithm 1 satisfies*

$$\begin{aligned} &\min_{k<K}\{V(\theta^*;\boldsymbol{\rho}) - V(\theta_i^k;\boldsymbol{\rho})\} \le \epsilon \\ &\text{if } K \ge \mathcal{O}\left(\frac{|S|^2|A|^2\sum_{j=1}^N\frac{1}{(1-\gamma_j)^6}}{(1-\sigma_2)^2\epsilon^2}\left\|\frac{d_{\boldsymbol{\rho}}^{\pi_{\theta^*}}}{\boldsymbol{\mu}}\right\|_\infty^2\right), \end{aligned} \quad (14)$$

where we denote $\left\|\frac{d_{\boldsymbol{\rho}}^{\pi_{\theta^*}}}{\boldsymbol{\mu}}\right\|_\infty = \max\limits_{\substack{s\in\mathcal{S}\\ j:s\in\mathcal{S}_j}}\frac{d_{j,\rho_j}^{\pi_{\theta^\star}}(s)}{(1-\gamma_j)\mu_j(s)}$.

Under Assumption 2, Algorithm 1 achieves a global optimality with the same rates as the ones in Agarwal et al. [2020], except for a factor $1/(1-\sigma_2)^2$ which captures the impact of communication graph $\mathcal{G}$. Eq. (14) also shows the impact of the initial distribution $\boldsymbol{\mu}$ on the convergence of the algorithm through the distribution mismatch coefficient. A bad choice of $\boldsymbol{\mu}$ may result in a local optimum (or stationary point) convergence by breaking Assumption 2, as we will illustrate by simulation in Section 4.2.



(a) No conflicts (b) Resolvable conflicts (c) Unresolvable conflicts (d) Start position top-left (e) Start position top-right

Figure 2: Evaluate Learned Policy in Multi-task GridWorld

5 EXPERIMENTAL RESULTS

5.1 GRIDWORLD PROBLEMS

In this section, we evaluate the performance of our proposed algorithm on two platforms: GridWorld and drone navigation. We first verify the correctness of our theoretical results by applying the decentralized policy gradient (DCPG) algorithm for solving small-scale GridWorld problems, where each agent uses a tabular policy. We next apply the proposed method to solve the more challenging problem of large-scale drone navigation in simulated 3D environments, where the policy is approximated by neural networks.

General setup. In each simulation, the agents runs a number of episodes of DCPG. In each episode, each agent computes its local gradient by using the Monte-Carlo method. Each agent then communicates with its neighbors over a fixed ring graph (i.e. agent i communicates with agent $i-1$ and $i+1$ for $i = 2, 3, ..., N-1$; agent 1 communicates with agent 2 and N; agent N communicates with agent $N-1$ and 1) and updates its iterates using (9). Given the communication graph $\mathcal{G}$, we generate the weight matrix W using the lazy Metropolis method [Olshevsky, 2014].

To illustrate the results in Theorem 2, we apply Algorithm 1 for solving the popular GridWorld problem under tabular settings, i.e., $\theta \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$. This is a notable small-scale RL problem, which can be solved efficiently by using tabular methods; see for example Sutton and Barto [2018, Example 4.1]. In this problem, the agent is placed in a grid of cells, where each cell can be labeled either by the desired goal, an obstacle, or empty. The agent selects an action from the set of 4 actions {up, down, left, right} to move to the next cell. It then receives a reward of $+1$ if it reaches the desired goal, -1 if it gets into an obstacle, and 0 otherwise. The goal of the agent is to reach a desired position from an arbitrary initial location in a minimum number of steps (or maximize its cumulative rewards).

For multi-task RL settings, we consider a number of different single GridWorld environments of size 10×10, where they are different in the obstacle and goal positions. We assign each agent to one environment. In this setting, the environments have the same transition probabilities. Therefore, Assumption 2 is satisfied when agents across environments use an identical initial state distribution.

For solving this multi-task GridWorld, the agents implement Algorithm 1 where the local gradients are estimated using a Monte-Carlo approach, and the states are their locations in the grid. After 1000 training episodes, the agents agree on a unified policy, whose performance is tested in parallel in all environments. The results are presented in Fig.2, where we combine all the environments into one grid. In addition, yellow and red cells represent the goal and obstacle, respectively. For each environment, we terminate the test when the agent reaches the goal or hits an obstacle. The light green path is the route which the agent visits in these environments. Since we have a randomized policy, we put the path mostly followed by the agents. Fig.2 (a)–(c) consider experiments on four environments, while (d) and (e) are on six environments.

In Fig.2(a), we illustrate the performance of the policy when there is no conflict between the environments, i.e., the block of one environment is not the goal of the others and vice versa. In this case, we can see that the algorithm returns an optimal policy which finds all the goals at the environments. Next, we consider the conflict setting in Fig.2(b), where one obstacle of environment 2 is the goal of environment 3. Here, the i number in white and black represents the goal and the obstacles of the i-th environment, respectively. Although in this case there is a conflict between the tasks, it is solvable, that is, the agents still can find the optimal policy to solve all the task. These simulations agree with our results in Theorem 2, which finds the global optimality.

We next consider an unsolvable conflict in Fig.2(c), where the goal of agent 2 is the obstacle of agent 3 and vice versa. In this case, there does not exists a policy that can always visit all goal positions without running into an obstacle. Instead, the agents need to make a compromise, where they finish three out of the four tasks.

To summarize, the experiments with no conflict and resolvable conflict have dynamics that allow the optimal value of (5) to be the sum of the optimal values of the individual tasks, while the experiment with unresolvable conflict does not. Nevertheless, in all three cases, DCPG successfully finds the global optimality of the global objective function (5).

Finally, we illustrate the impact of the initial conditions with

Table 1: MSF of Learned Policy

Policy (REINFORCE)	Env0	Env1	Env2	Env3	Sum
SA-0	15.9 ± 5.3	4.5 ± 1.2	4.1 ± 1.3	3.6 ± 3.0	28.1
SA-1	3.0 ± 0.2	55.4 ± 29.3	9.7 ± 2.8	8.1 ± 3.8	76.2
SA-2	1.5 ± 0.5	0.8 ± 0.2	21.1 ± 18.3	2.0 ± 0.6	25.4
SA-3	2.3 ± 0.5	0.8 ± 0.2	8.6 ± 2.0	40.1 ± 17.4	51.8
DCPG (proposed)	**25.2 ± 20.1**	**67.9 ± 35.5**	**40.5 ± 18.0**	**61.8 ± 39.2**	**195.4**
Policy (A2C)	Env0	Env1	Env2	Env3	Sum
SA-0	21.8 ± 6.5	7.0 ± 0.8	15.1 ± 5.4	14.9 ± 8.2	58.8
SA-1	1.3 ± 0.4	**54.1 ± 20.1**	2.8 ± 0.9	6.4 ± 1.2	59.4
SA-2	1.8 ± 0.7	3.9 ± 0.3	105.2 ± 38.5	9.9 ± 1.3	120.8
SA-3	1.1 ± 0.2	1.4 ± 0.2	15.8 ± 5.0	78.6 ± 25.9	96.9
DCPG (proposed)	**25.2 ± 7.5**	50.1 ± 24.6	**165.8 ± 64.6**	**159.6 ± 61.0**	**380.7**
Policy (PPO)	Env0	Env1	Env2	Env3	Sum
SA-0	**28.3 ± 15.5**	11.2 ± 6.3	8.7 ± 5.9	13.5 ± 5.7	61.7
SA-1	1.1 ± 0.6	**75.3 ± 43.2**	1.6 ± 0.4	1.6 ± 0.8	79.6
SA-2	2.5 ± 1.8	3.0 ± 1.1	63.2 ± 36.4	15.6 ± 10.6	84.3
SA-3	1.9 ± 1.6	1.2 ± 0.5	14.3 ± 8.7	139.0 ± 72.5	156.4
DCPG (proposed)	26.3 ± 10.9	66.7 ± 30.8	**144.0 ± 82.4**	**195.2 ± 92.4**	**432.2**

the simulations in Fig.2(d) and (e). In (d), if the agents start from the top left corner they cannot find the optimal solution. However, when the agents start from the top right corner the algorithms return the gobal optimality as shown in (e). This empirical evidence hints that to achieve the global optimality with the DCPG algorithm, conditions on the initial state distribution like Assumption 2 may be necessary.

5.2 DRONE NAVIGATION

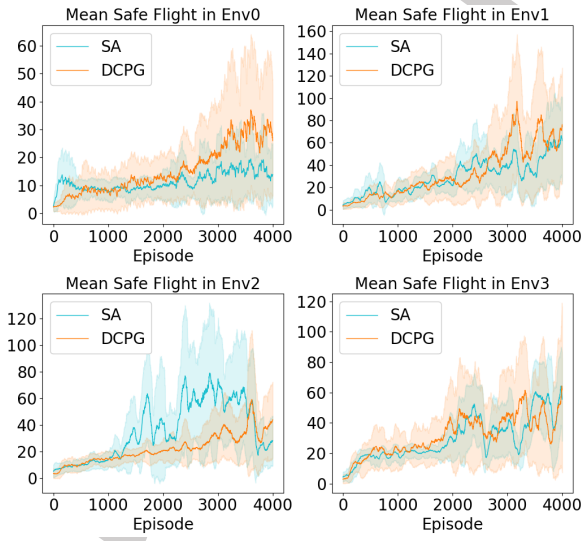


Figure 3: MSF During Training (REINFORCE)

For the drone experiment we use PEDRA, a 3D stimulated drone navigation platform [PED]. In this platform, a drone agent is equipped with a front-facing camera, and takes actions to control its flight. The reward received by the drone agent is designed to encourage the drone to stay away from obstacles. We select 4 indoor environments on the PEDRA platform (denoted as Env 0-3), which contain widely different lighting conditions, wall colors, furniture objects, and hallway structures. The performance of a policy is quantified by the mean safe flight (MSF), the average distance travelled by the agent before it collides with any obstacle. This is a standard criterion in evaluating the performance of flying autonomous vehicles [Sadeghi and Levine, 2016].

Instead, to evaluate the policy learned using Algorithm 1 (DCPG), we compare it with the single agent trained independently in each environment. For brevity, we denote by SA-i the single agent trained in environment i. We note that the SAs can be considered as the solutions to the local objective functions, while DCPG optimizes the sum of the local objective functions. Therefore, if trained to the global optimum, each SA provides an upper bound on the performance of the DCPG policy in the respective environment. The aim of the experiments is to show in practical problems, the DCPG policy often performs close to this bound.

To demonstrate the compatibility of our algorithm with a wide range of policy gradient methods, we conduct three sets of experiments, where we run Algorithm 1 with the gradient g_i^k estimated by three popular variants of policy gradient algorithms: REINFORCE, advantage actor-critic (A2C), and proximal policy optimization (PPO). In each case, a 5-layer neural network is used to approximate the

policy[4]. We stress that in each set of the experiments, the SAs and DCPG are trained identically, with the only difference being whether the agents communicate their policies.

In Fig.3, we show MSF of the DCPG and SA policies in the training phase with the REINFORCE algorithm.

Finally, we test the policies learned by DCPG and SAs in the four environments. The results are presented in Table 1. Across the three sets of experiments, we consistently see the performance difference between DCPG and the SAs. As expected, SA-i only performs well in i-th environment but does not generalize to environment it has not seen. On the other hand, the policy returned by DCPG performs very well in all environments. Surprisingly, DCPG often performs even better than each SA-i in the i-th environment, which we observe is due to the benefits of learning common features and representation among the agents.

6 CONCLUSION

By combining consensus optimization with the policy gradient algorithm, we propose a decentralized method that aims to learn a unified policy in MTRL problems. We theoretically show that the convergence of multi-task algorithm achieves the same convergence rate as the single task algorithm within constants depending on the connectivity of the network, and support our analysis with a series of experiments. Future directions from this work include investigating the possibility of relaxing Assumption 2 in achieving the global optimality and improving the convergence rate.

[4]Implementation details are presented in Section C.1 of the supplementary material.

References

https://icsrl.ece.gatech.edu/pedra.

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. volume 125 of *Proceedings of Machine Learning Research*, pages 64–66, 2020.

A. Anwar and A. Raychowdhury. Autonomous navigation via deep reinforcement learning for resource constraint edge nodes using transfer learning. *IEEE Access*, 8:26549–26560, 2020.

Mahmoud Assran, Joshua Romoff, Nicolas Ballas, Joelle Pineau, and Michael Rabbat. Gossip-based actor-learner architectures for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 13320–13330, 2019.

T. Chu, S. Chinchali, and S. Katti. Multi-agent reinforcement learning for networked system control. In *International Conference on Learning Representations (ICLR)*, 2020.

Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Sharing knowledge in multi-task deep reinforcement learning. 2020.

D. Ding, X. Wei, Z. Yang, Z. Wang, and M.R. Jovanović. Fast multi-agent temporal-difference learning via homotopy stochastic primal-dual optimization. available at: `https://arxiv.org/abs/1908.02805`, 2019.

T. T. Doan, S. T. Maguluri, and J. Romberg. Finite-time analysis of distributed TD(0) with linear function approximation on multi-agent reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1626–1635, 2019.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. volume 80 of *Proceedings of Machine Learning Research*, pages 1407–1416, 2018.

Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. available at: `https://arxiv.org/abs/1812.05905`, 2019.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3796–3803, 2019.

S. Kar, J. M. F. Moura, and H. V. Poor. Qd-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus + innovations. *IEEE Trans. Signal Processing*, 61:1848–1862, 2013.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016.

Guanghui Lan. Policy mirror descent for reinforcement learning: Linear convergence, new sampling complexity, and generalized problem classes. *arXiv preprint arXiv:2102.00135*, 2021.

D. Lee, N. He, P. Kamalaruban, and V. Cevher. Optimization for reinforcement learning: From single agent to cooperative agents. available at: `https://arxiv.org/abs/1912.00498`, 2019.

W. Li, B. Jin, X. Wang, J. Yan, and H. Zha. F2a2: Flexible fully-decentralized approximate actor-critic for cooperative multi-agent reinforcement learning. available at: `https://arxiv.org/abs/2004.11145`, 2020.

Lydia T Liu, Urun Dogan, and Katja Hofmann. Decoding multitask dqn in the world of minecraft. In *The 13th European Workshop on Reinforcement Learning (EWRL) 2016*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Arun Nair, Praveen Srinivasan, Sam Blackwell, et al. Massively parallel methods for deep reinforcement learning. 07 2015.

Alex Olshevsky. Linear time average consensus on fixed graphs and implications for decentralized optimization and multi-agent control. *arXiv preprint arXiv:1411.4186*, 2014.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, et al. Dota 2 with large scale deep reinforcement learning. 2019. URL `https://arxiv.org/abs/1912.06680`.

Kirill Ovchinnikov, Anna Semakova, and Alexey Matveev. Decentralized multi-agent tracking of unknown environmental level sets by a team of nonholonomic robots. In *2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 352–359. IEEE, 2014.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994.

G. Qu and N. Li A. Wierman. Scalable reinforcement learning of localized policies for multi-agent networked systems. available at: `https://arxiv.org/abs/1912.02906`, 2019.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

David Silver, Aja Huang, Chris J Maddison, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning, 2nd Edition*. MIT Press, 2018.

René Traoré, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Guanghang Cai, Natalia Díaz-Rodríguez, and David Filliat. Discorl: Continual reinforcement learning via policy distillation. *arXiv preprint arXiv:1907.05855*, 2019.

H-T Wai, Z. Yang, Z. Wang, and M. Hong. Multi-agent reinforcement learning via double averaging primal-dual optimization. In *Annual Conference on Neural Information Processing Systems*, pages 9672–9683, 2018.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Chao Yu, Jiming Liu, and Shamim Nemati. Reinforcement learning in healthcare: a survey. *arXiv preprint arXiv:1908.08796*, 2019.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. available at: `https://arxiv.org/abs/2001.06782`, 2020.

Sihan Zeng, Thinh T Doan, and Justin Romberg. Finite-time analysis of decentralized stochastic approximation with applications in multi-agent and multi-task learning. *arXiv preprint arXiv:2010.15088*, 2020.

K. Zhang, Z. Yang, and T. Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. available at: `https://arxiv.org/abs/1911.10635`, 2019.

Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. Fully decentralized multi-agent reinforcement learning with networked agents. volume 80 of *Proceedings of Machine Learning Research*, pages 5872–5881, 2018.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE, 2017.